

**AUTOMATED DETECTION AND COUNTING OF PEDESTRIANS ON AN
URBAN ROADSIDE**

A Thesis Presented

by

GAYATRI D. PRABHU

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2011

ELECTRICAL AND COMPUTER ENGINEERING

© Copyright by Gayatri D. Prabhu 2011

All Rights Reserved

**AUTOMATED DETECTION AND COUNTING OF PEDESTRIANS ON
AN URBAN ROADSIDE**

A Thesis Presented

by

GAYATRI D. PRABHU

Approved as to style and content by:

Russell G. Tessier, Chair

Daiheng Ni, Member

Patrick A. Kelly, Member

C. V. Hollot, Department Head
Electrical and Computer Engineering

ACKNOWLEDGMENTS

I thank the Massachusetts Department of Transportation-Highway Division for supporting my thesis work. I am grateful to Prof. Russell Tessier, my committee chair, for giving me an opportunity to work on a challenging project. I also extend my gratitude to Prof. Daiheng Ni and Prof. Patrick Kelly for being on my thesis committee. I acknowledge the hardware support provided by Cheng Zhang, Gilbert Kim and David Glazier of the Civil Engineering Department. I deeply appreciate the contributions and immense zest of Chaoqun (Enzo) Jia who helped me achieve the required goals in time. I would also like to make use of this opportunity to thank my fellow RCG members for sportingly agreeing to be “pedestrians” while testing the software at various stages, especially Murtaza for his insightful discussions and for helping me procure data. Last, but not the least, I thank my family for providing me encouragement and support to pursue higher studies in the United States. Special thanks to my husband, Abhishek Kamath, who encouraged me when the going got tough and helped me stay focused and motivated throughout the duration of the project.

ABSTRACT

AUTOMATED DETECTION AND COUNTING OF PEDESTRIANS ON AN URBAN ROADSIDE

SEPTEMBER 2011

GAYATRI D. PRABHU

B.Tech, NATIONAL INSTITUTE OF TECHNOLOGY, CALICUT, INDIA

M.S. E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Russell G. Tessier

Transportation planning, which involves the development of facilities to accommodate pedestrian and bicycle traffic, demands a mechanism for the collection of accurate pedestrian statistics. This thesis implements an automated system that detects and counts pedestrians with 85% accuracy. Two approaches have been considered and evaluated for the purpose of collecting pedestrian traffic statistics. The first approach employs the Autoscope Solo Terra, a traffic camera which is widely used to monitor vehicular traffic. The Solo Terra supports an image processing-based detector that counts the number of objects crossing user-defined areas in the captured image. The count is updated based on the amount of movement across the selected regions. The approach, however, is incapable of discerning whether the movement was caused by a pedestrian or a vehicle. Therefore, a second approach has been considered that uses a histogram of oriented gradients (HoG), an advanced vision based algorithm proposed by Dalal et al. which is capable of distinguishing a pedestrian from a non-pedestrian based on an omega shape formed by the head and shoulders of a human being. The implemented detection

software processes video frames that are streamed from a low-cost digital camera. The frames are divided into sub-regions which are scanned for an omega shape whenever movement is detected in those regions. The two approaches have been evaluated in terms of count accuracy, cost and ease of deployment. It has been found that the HoG-based approach degrades in performance due to occlusion under dense pedestrian traffic conditions whereas the Solo Terra approach appears to be more robust. Undercounts were encountered using the Solo Terra approach when a sidewalk is fully occupied across its width. Overcounts caused by movements of overhead tree branches, wires and shadows of pedestrians in the Solo Terra approach also reduced count accuracy under certain weather conditions. To combat the disadvantages of both the approaches, they were integrated to form a single system where count is incremented predominantly using the Solo Terra. The HoG-based approach corrects the obtained count under certain conditions. A preliminary prototype of the integrated system has been verified.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	iv
ABSTRACT	v
LIST OF TABLES	ix
LIST OF FIGURES	x
 CHAPTER	
1. INTRODUCTION	1
2. BACKGROUND AND RELATED WORK	4
2.1. Existing methods for candidate generation	4
2.1.1. Doppler Effect based techniques	4
2.1.2. Computer vision-based techniques	5
2.1.3. Miscellaneous techniques	8
2.2. Pedestrian classification	9
2.2.1. Motion-based classification	10
2.2.2. Shape-based classification	10
2.3. Pedestrian tracking	17
2.4. Miscellaneous techniques	20
2.4.1. Zone-based detection	20
2.4.2. Vertical projection based methods	21
2.5. Summary	21
3. AUTOSCOPE SOLO TERRA	23
3.1. Solo Terra camera-based system	23
3.1.1. Autoscope Solo Terra	24
3.1.2. Autoscope Software Suite	24
3.1.3. Terra Interface Panel	24
3.2. Components of the Autoscope Software Suite	25
3.2.1. Autoscope Configuration Wizard	25
3.2.2. Autoscope Detector Editor	26
3.2.3. Autoscope Data Collector	26
3.2.4. Autoscope Video Player	27
3.2.5. Autoscope Property Editor	27
3.2.6. Autoscope Software Installer	27
3.2.7. Autoscope Field of View Calculator	27
3.3. Detection capabilities	28
3.3.1. Detection zone	31

3.3.2. Detectors	31
4. SYSTEM DESIGN USING THE AUTOSCOPE SOLO TERRA	43
4.1. Design of the hardware support structure for the Solo Terra.....	43
4.2. Software resources used for pedestrian detection	45
4.3. Detector configuration for the Solo Terra.....	45
4.4. Pedestrian counting algorithms using the Solo Terra	48
4.4.1. State averaging algorithm	49
4.4.2. State matrix approach	50
4.4.3. M of N approach	51
5. VISION BASED SHAPE DETECTION	52
5.1. Overview.....	52
5.2. Background determination and subtraction	53
5.3. Classification into pedestrians and non-pedestrians	55
5.3.1. Representation of shapes using HoGs.....	56
5.3.2. Classification of shapes.....	64
5.4. Counting algorithm	70
5.5. Software libraries	71
6. INTEGRATED SYSTEM FOR ACCURATE COUNTING.....	72
6.1. The need for an integrated approach.....	72
6.2. Experimental framework.....	75
7. EXPERIMENTS AND RESULTS	78
7.1. Experiments using the Autoscope Solo Terra.....	78
7.1.1. Results from the state averaging approach	80
7.1.2. Results from the state matrix approach.....	84
7.1.3. Results from the M of N approach.....	84
7.1.4. Effect of background refresh rate	86
7.2. Experiments using the low-cost camera	86
7.2.1. Evaluation of HoG parameters.....	88
7.2.2. Experiments with SVM classifier	89
7.2.3. Software performance issues	93
7.3. Results of integrating the two approaches	95
8. CONCLUSION AND FUTURE WORK.....	99
BIBLIOGRAPHY	101

LIST OF TABLES

Table	Page
1. Biometric distances [34]	13
2. Summary of different pedestrian detection techniques.....	22
3. Traffic data collected by the Solo Terra camera [61]	30
4. Type and functionalities of available detectors in Solo Terra [61].....	32
5. Boolean operations supported by Autoscope Software Suite [61]	38
6. Accuracy for each of the averaging parameter m	81
7. Best and worst case results for 2 x 15 configuration with uniform spacing	82
8. Accuracy for different pedestrian densities	82
9. Best and worst case results for a 2 x 15 configuration with non-uniform spacing	84
10. Results of M of N approach	85
11. Optimum values of parameters for " Ω " detection	89
12. Execution time reported by gprof for initial software	94
13. Execution time reported by gprof	95
14. Preliminary results for the integrated approach	97

LIST OF FIGURES

Figure	Page
1. Stages in the pedestrian counting process.....	3
2. Microwave RADAR technique [4]	5
3. Pedestrian detection using range information.....	7
4. Images obtained from an infrared camera [19].....	8
5. Rhythmic motion of pedestrians [24]	10
6. " Ω " template with normals [12].....	12
7. Circular curve templates [33].....	12
8. Biometric measurements for golden ratio [34]	13
9. Wavelet coefficients at different scales [36].....	14
10. Template hierarchy proposed in [13].....	15
11. A thresholded image and its vertical projection histogram [57].....	21
12. The Solo Terra camera based system.....	24
13. Terra Interface Panel [59]	25
14. Screenshot of the Autoscope Field of View calculator.....	28
15. Anomaly due to non-uniform background.....	33
16. Jagged detector - The detector appears like a stair of steps	33
17. Illustration of presence detectors.	36
18. Count detector as a series of white lines that cross the intersection [61]	36
19. Illustration of a speed detector.	37
20. ORing of two directional presence detectors	39
21. Local contrast detectors	39

22. Detector Stations and Label detectors.....	41
23. Mounting structure along with the raw materials	44
24. Activation of detectors by pedestrians	46
25. Multiple detectors turning ON due to irregular movements of pedestrians.....	48
26. Example of polled data information	49
27. Detector configuration used in the state averaging approach	50
28. Detector configuration used for the state matrix approach	50
29. Detector configuration for the M of N approach	51
30. Procedure followed for the automated detection and counting of pedestrians based on histograms of gradients (HoGs)	53
31. Flowchart summarizing the background determination process	55
32. Dense overlapping windows for HoG calculation	57
33. Original image along with the detected edges	58
34. Block histogram with 8 bins and 4 cells	60
35. Images with different values of gamma (Source:wikipedia)	61
36. Flowchart representing the process of HoG calculation	63
37. Hyperplanes separating the two categories.....	65
38. Representation of SVM [65]	67
39. Generation of positive samples for training.....	69
40. Negative samples in INRIA dataset [68]	69
41. Detector transition times for the Solo Terra when a pedestrian walks across the detector region	70
42. (a) Streaking (b) Blooming (c) Simultaneous blooming and streaking	73
43. Detectors remaining ON due to blooming resulting in under-counts	74
44. Experimental framework for the integrated approach	77

45. Test locations for counting pedestrians using Solo Terra	79
46. Detectors counting pedestrians	80
47. A five column detector configuration for averaging approach	83
48. Detector configuration with non-uniform spacing.....	83
49. A detector configuration to find the effect of sizes on pedestrian count accuracy	86
50. Scanning for omega shapes in user-selected frame regions.....	87
51. Original frame; Results of background subtraction - Identified foreground is white in color, the black pixels indicate the background.....	88
52. Omega detection results on the MIT Pedestrian Dataset [66]	90
53. (a) Linear SVM where a hyperplane divides the two categories (b) Gaussian SVM where curves separate the two categories	91
54. Histogram of a positive sample and a non-omega shape	92
55. Pedestrian detection using the low-cost camera.	92
56. Results of rectangle grouping	93
57. Regions demarcated for searching for omega shapes.	96
58. Scenario where count gets corrected by the HoG approach in the integrated system	97
59. Scenarios which invoked omega detection algorithm but did not increment count.....	97
60. Scenario where omega detection does not get invoked	98
61. Effect of window sliding distance on accuracy. (a) Sliding distance = 8 pixels, (b) Sliding distance = 6 pixels	98

CHAPTER 1

INTRODUCTION

Every year pedestrian fatalities constitute around 12 percent of all traffic fatalities causing approximately 4,000 deaths and 59,000 injuries [1]. The fatalities are more frequent in urban areas due to a higher volume of pedestrians than in rural areas. For safe accommodation of pedestrian and bicycle traffic, transportation planning requires an accurate estimate of the occupancy of walkways and bike lanes. Hiring human resources to count pedestrians at various locations at different times of the day over a long period is a cost-ineffective solution. The need to explore automated techniques that detect and count pedestrians stems from the following demands.

- Economical collection of data pertaining to bicycle and pedestrian traffic which is required for transportation planning.
- Alerting drivers to pedestrians in the vicinity of vehicles for accident avoidance.

A methodology ideal for alerting systems may not be suitable for estimating pedestrian volume. This thesis focuses on providing a cost-effective solution for pedestrian counting to aid transportation planning. The term pedestrian encompasses upright people, people in wheelchairs and people on skateboards. The objective of the thesis was to implement an automated system for efficient, economical and accurate collection of pedestrian and bicyclist traffic data. An automated counting system can be deployed on a wide scale only if the system provides a pedestrian count with at least 85% accuracy.

Among the prevailing techniques, computer vision-based methods are suitable for counting with 85% accuracy. Two approaches are considered to detect and count pedestrians, a zone-based approach using an Autoscope camera [2] and a vision-based shape detection approach using a low-cost camera. The first approach makes use of a traffic camera that is typically used for vehicle monitoring. The configuration of the inbuilt software is modified to suit pedestrian counting requirements. The vision-based shape detection approach scans for a pedestrian shape in an image and makes decisions based on advanced machine learning algorithms. The scope of the thesis includes an evaluation of the two approaches for pedestrian counting in terms of accuracy, cost and ease of deployment.

Pedestrian detection presents challenges caused by human articulations and outdoor environmental conditions due to weather and lighting. It is very difficult to detect pedestrians in various poses, angles and clothing. Pedestrian traffic is highly irregular with movements in random directions and multiple entry and exit points in a frame. Occlusion, which refers to obscuration from view, presents a major challenge. The camera used to provide detection algorithm input should be mounted at a suitable height and angle to capture clear images of pedestrians. Variable lighting conditions, the presence of shadows and camera position should be considered during detection algorithm design.

A pedestrian counting algorithm consists of three stages [3]: candidate generation, pedestrian classification, and pedestrian tracking, as shown in Figure 1. The candidate generation stage involves the capturing of images or relevant data of potential pedestrian candidates from the field of interest through shape, motion and distance cues. The

pedestrian classification stage identifies pedestrians from the generated candidates using machine learning algorithms. The tracking stage traces the identified pedestrians until they disappear from the field of view. Often the line between the three stages is blurred and the stages are merged in some cases.

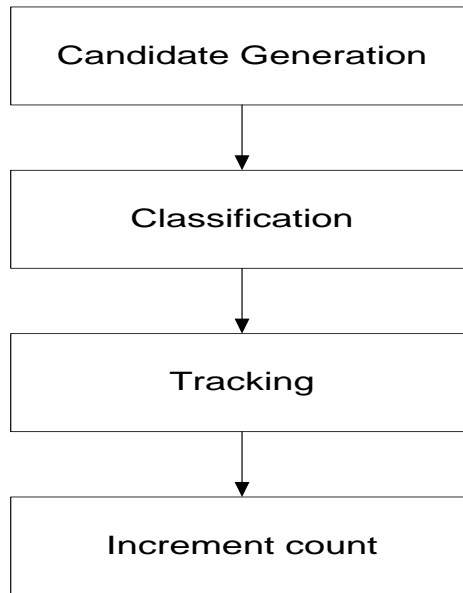


Figure 1: Stages in the pedestrian counting process

The document is organized as follows. Chapter 2 provides a detailed literature survey on prevailing techniques for pedestrian detection and counting. Chapter 3 describes the capabilities of the Autoscope Solo Terra, the traffic camera deployed for vehicle traffic monitoring. Chapter 4 describes the implementation details of the Autoscope camera-based approach. Chapter 5 explains the methodology adopted to implement a low-cost camera system. Chapter 6 discusses the preliminary prototype of an integrated approach to combat the disadvantages of the above approaches. Chapter 7 presents the results of the experiments conducted as part of the thesis. Chapter 8 concludes the thesis and provides directions for future work.

CHAPTER 2

BACKGROUND AND RELATED WORK

Various pedestrian detection and counting techniques have been developed over the years. This chapter provides a survey of prevailing techniques.

2.1. Existing methods for candidate generation

Candidate generation, the first step in pedestrian detection, involves the identification of potential pedestrians and the generation of candidates in the form of images. The earliest pedestrian identification techniques were Doppler Effect-based techniques [4], which involved signal wave transmission. Presently, most candidate generation techniques are based on computer vision and require the processing of pixelated images. Vision-based methods are appropriate for counting purposes since they address some of the disadvantages of Doppler Effect based techniques. The following sections outline the principles behind some candidate generation methods.

2.1.1. Doppler Effect based techniques

A Doppler Effect-based detector continuously transmits waves of a constant pre-determined frequency. Any object passing through the transmitted beam causes reflections that introduce a shift in the measured frequency. This shift, termed the Doppler Effect, is analyzed at the transmitter to determine the presence of an object. Currently electromagnetic waves (microwaves) [4][5][6][7], ultrasonic sound waves [6][7] and active infrared beams [6][8] are used for transmission. A Doppler Effect-based system is illustrated in Figure 2. Even though a shift in frequency indicates the presence of an object, it does not provide sufficient information to determine whether the object is

a pedestrian. The beam must be accurately targeted if the object to be detected (pedestrian) is smaller than the surrounding objects (moving vehicles).

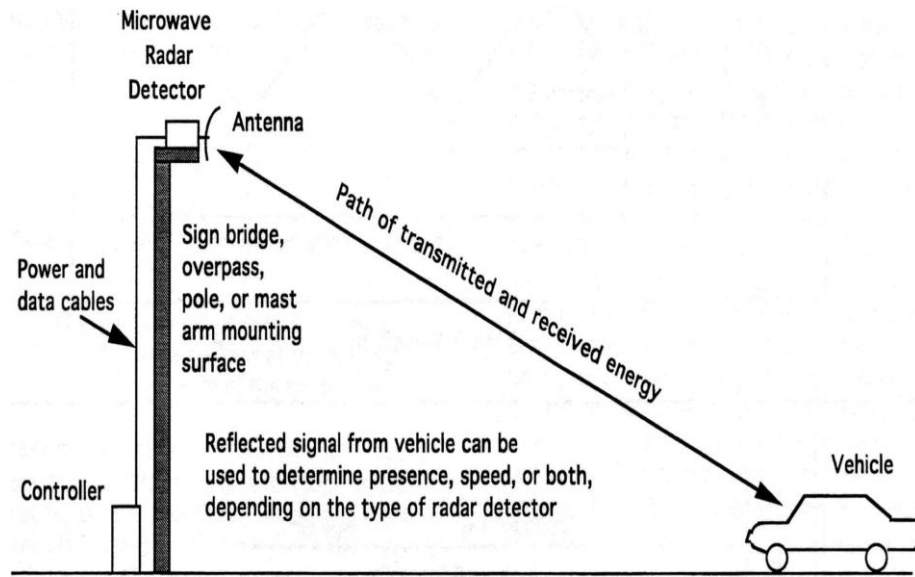


Figure 2: Microwave RADAR technique [4]

2.1.2. Computer vision-based techniques

Computer vision based techniques employ images or videos obtained from a lens-based camera to single out objects that are likely to be pedestrians. The simplest approach to extract information about pedestrian candidates is through background subtraction, the process of removing background information from an image. Objects extracted from the resulting foreground are passed as inputs to the classification stage.

Numerous methods have been proposed to identify background pixels. Moeslund et al. [9] consider an image background to be the image of a scene captured without foreground objects. Sexton et al. [10] consider the background to be the pixels whose values remain fixed for a particular number of frames. These methods are inaccurate as lighting conditions may vary the values of image pixels.

Statistical measures are widely used to reduce the effects of pixel value variations caused by lighting conditions. Examples of such measures include the minimum, maximum and largest inter-frame absolute difference for each pixel found during the training period [11], the mean or median value of a pixel by a unimodal distribution similar to a Gaussian distribution [12] and the variance of a pixel modeled by a mixture of Gaussians using non-parametric models [13][14]. It is necessary to periodically refresh the background pixels determined by these methods to reflect the changes induced by lighting conditions.

A codebook-based algorithm recommended by Kim et al. [15] encodes the background on a pixel-by-pixel basis. The proposed algorithm adopts a quantization and clustering approach [16]. Samples at each pixel are clustered into a set of codewords. Each pixel has a different codebook size based on its sample variation. A codebook entry consists of two vectors representing the RGB color values along with brightness and temporal values (for example, the minimum and maximum values to account for lighting conditions). During training, each sample value is expressed in terms of a brightness measure and a color distortion measure. The formulated vectors are compared against the current codebook entries to determine an encoding approximation. If the difference between the codebook entry and the sample lies below a certain threshold, the codebook entry is chosen as the approximate encoding. An exhaustive search is not conducted to find the entry with the lowest difference; rather the first codeword which satisfies the threshold condition is chosen as the approximation. A new codeword is created in the absence of an approximation. The same procedure is followed to periodically update the codebook. After training, the color and brightness vectors of pixel values are compared

against the codebook entries. If no match is found, a pixel is considered to be a foreground pixel.

Augmenting pixel value information with additional information such as image gradients and edge features [10] improves the accuracy of background determination. Recently, range information has been recognized as a powerful cue to distinguish foreground from background [17]. An image is generated whose intensity values depend on the distance of the object from the camera and the perceived depth of the object. Hence, brighter values represent a shorter distance, as demonstrated by Figure 3. The quadratic decay of range resolution with distance can be assessed using the polar perspective map proposed by Howard et al. [18]. The obtained stereo image is subsequently morphed to remove noise and smooth foreground images.

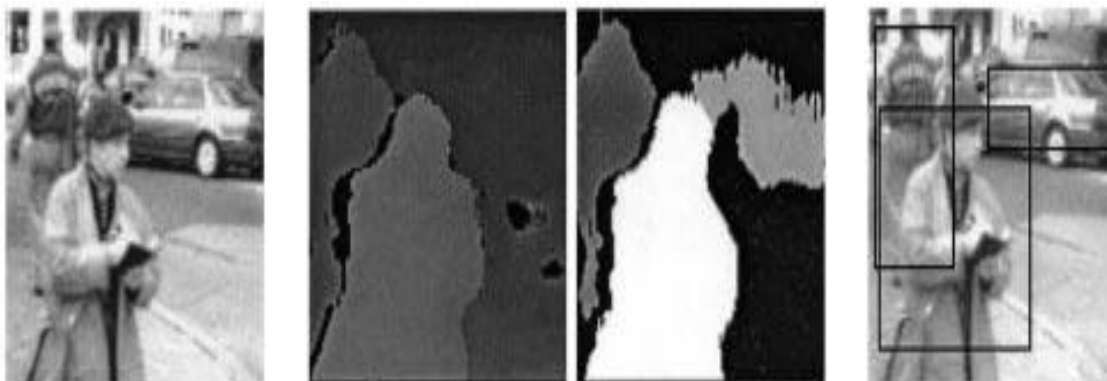


Figure 3: Pedestrian detection using range information
Left to Right - Image from stereo camera; Perceived depth map; Potential pedestrians; Detected objects marked by boxes. [17]

Although range information provides robustness against lighting conditions, it is difficult to distinguish between objects with the same texture and objects that are at the same depth as background regions, for instance, a person standing against a wall. Hence, background subtraction cannot be carried out using range information alone.

2.1.3 Miscellaneous techniques

Passive infrared sensors and laser scanners are also used to generate images from the field of interest containing potential pedestrians. The generated images are then subjected to the same classification process used by computer vision-based techniques.

Passive infrared sensors [19][20] generate grey level images based on the heat emitted by human body. The intensity of a pixel corresponds to the temperature of the target object. Figure 4 illustrates an image obtained from an infrared camera. Although the approach is robust for a variety of lighting conditions, it can be inaccurate due to the error rate caused by heat emitted from clothing worn by pedestrians. Furthermore, the system does not efficiently detect still pedestrians.



Figure 4: Images obtained from an infrared camera [19]

Laser scanner systems used for candidate generation are based on a time-of-flight principle [21]. A 2D rotating prism is used to transmit laser pulses. These pulses, which are switched on for a very short duration, illuminate the scene and are reflected by objects. The camera lens gathers the reflected light and projects it onto the sensor plane. Each photo-detector pixel in the sensor plane is connected to a counter running at several

gigaHertz, which stops counting when light is sensed. The pixel value therefore depends on the value of the counter. More distant objects are projected with lower intensity pixels due to the longer duration required for laser pulse reflection and sensor region activation. The images formed by laser scanner thus provide range information. The pulse width of the transmitted laser determines the maximum range of operation. However, complex signal processing steps hinder the deployment of laser systems.

Ultra-Wideband (UWB) [7][22] is another emerging technology for pedestrian counting. Radio pulses of a wide frequency range are transmitted over a short distance and pedestrian presence is determined from the reflected waves. The wide bandwidth of UWB imparts robustness to interference caused by weather conditions leading to the additional advantage of reduced transmission power. The inherent precision timing of pulses, development of advanced timers and the potential to support high data rate communication for real-time operations make UWB a promising technique for detection.

Nissan, a car manufacturer, utilizes cellular phone signals to detect pedestrians [23]. The ITS system proposed by the automobile giant alerts drivers to pedestrians by processing the location data transmitted by a pedestrian's cellular phone signals and GPS data pertaining to the position of the car. This technology is not suited for counting since the presence of a pedestrian does not always imply the presence of cellular signals.

2.2. Pedestrian classification

The processed information obtained from candidate generation acts as the input to classification, the second stage in the counting process. The classification stage categorizes candidates as pedestrians or non-pedestrians based on their shape or motion.

In some cases, the classification techniques can also act as candidate generators. For instance, motion-based classifiers eliminate a separate candidate generation process.

2.2.1. Motion-based classification

The rhythmic gait of pedestrians can be utilized to distinguish them from other objects [24]. Mori et al. [25] detect pedestrians through a maximum entropy method by examining the periodic changes in image intensity due to walking. Cutler et al. [26] applied Fourier transformation with Hanning windows to detect gait periodicity and to exploit the strong correlation between pedestrian cadence and stride length. The motion-based approach is not popular since it requires a pedestrian's legs to be visible to detect rhythmic gait, as shown in Figure 5. This method is also limited by a failure to detect stationary pedestrians and a higher processing time requirement for procuring a sequence of images.

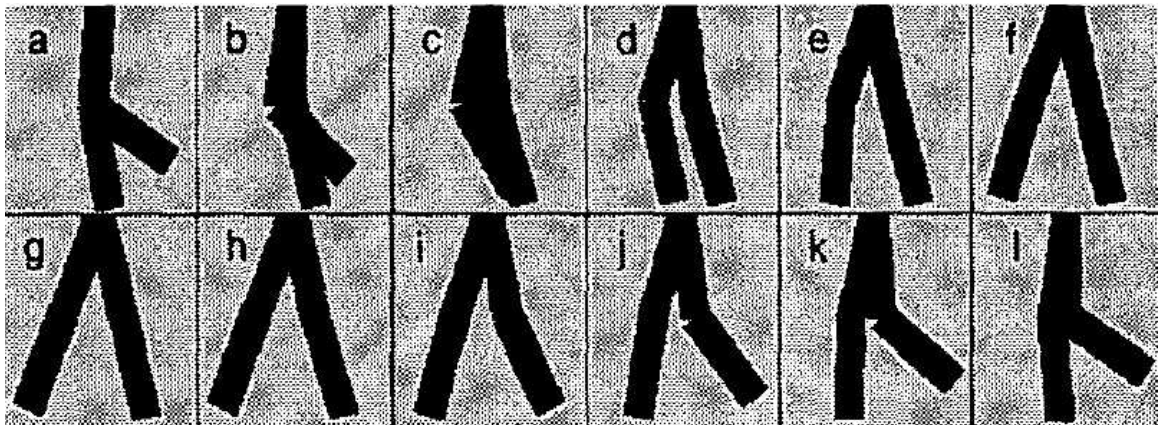


Figure 5: Rhythmic motion of pedestrians [24]

2.2.2. Shape-based classification

Pedestrians can also be identified by their shape. Even though it is difficult to accommodate the various poses and articulations of human beings, shape-based classification is popular since it requires limited processing time compared to motion-

based classification. The following three aspects need to be considered when designing a shape-based classification approach [27].

- Shape selection should be unique to the object that must be detected.
- A robust shape representation that is insensitive to object variations in scale, orientation and translation is needed.
- A decision-making algorithm to determine whether the shape belongs to the object under consideration is needed

The following sections highlight research work addressing these three aspects.

2.2.2.1. Selection of a shape that is unique to the object

Several researchers have proposed the full human body shape as the unique identifier, although some have proposed part-based classification. Snidaro et al. [28] suggest mounting a camera so that the sizes of all pedestrians are relatively constant. Hence, counting the number of pedestrians in a group involves a simple operation of dividing a larger blob by the average area of a person [28]. A head: torso ratio of 1 to 4 can be considered to be a pedestrian [29].

Park et al. [30] developed a human model built on a group of strong local convex shape descriptors. Human body parts such as heads, torsos, arms and legs are represented as convex shapes defined by center locations and the scale of the circle that best approximates the shape contour. The distribution of convex shapes in a given frame forms the basis for classifying objects as pedestrians or non-pedestrians.

A strong characteristic shape belonging to pedestrians is the “ Ω ” shape formed by a head and shoulders [31]. The vertical symmetry and presence of strong edges

contributing to an “ Ω ” shape can be utilized to detect pedestrians. Pedestrians can also be identified by calculating the relative position of the arms, legs and head [32].

2.2.2.2. Representation of shapes

The simplest representation of a shape is the values of pertinent geometric parameters. Zhao et al. [12] built the “ Ω ” template by considering the human body as a collection of ellipsoids with each ellipsoid characterized by its length and fatness. The contours for the “ Ω ” template are generated from the projected ellipses. Normals, as shown in Figure 6, are computed at contour points to determine the orientation of Ω .

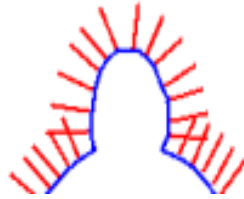


Figure 6: “ Ω ” template with normals [12]

Circular curve templates, shown in Figure 7, are used to detect people from overhead views [33].

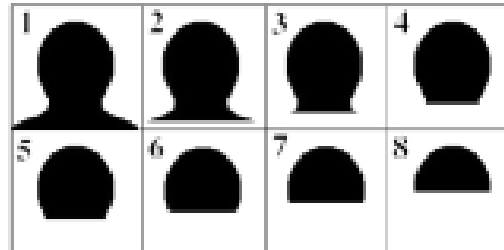


Figure 7: Circular curve templates [33]

The need for a common model to account for all pedestrian sizes ranging from adults to children motivates the use of the golden ratio [34][35] of biometrics for classification. The ratio of height and width measurements of various parts of the body, indicated in Figure 8, amount to the golden ratio $\Phi = 1.618$. The parameters listed in

Table 1 are measured for all generated candidates and the ratios are compared against the golden reference values.

$$\frac{|aj|}{|ai|} = \frac{|ai|}{|ah|} = \frac{|ah|}{|ag|} = \phi \quad \text{and} \quad \frac{|kp|}{|lo|} = \frac{|lo|}{|mn|} = \phi$$

Table 1: Biometric distances [34]

Distance	Meaning
aj	Height of the human body
ac	Distance from the top of the head to the forehead
ad	Distance from the top of the head to the eyes
mn	Width of the head
af	Distance from the top of the head to the base of the skull
lo	Width of the shoulders
ah	Distance from the top of the head to the navel and the elbows

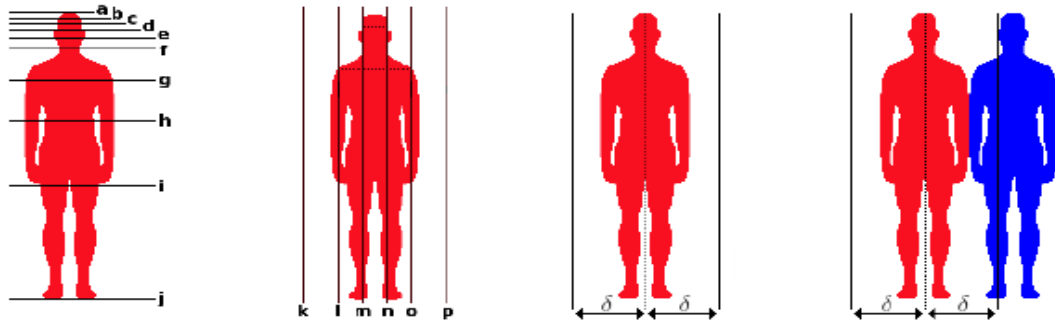
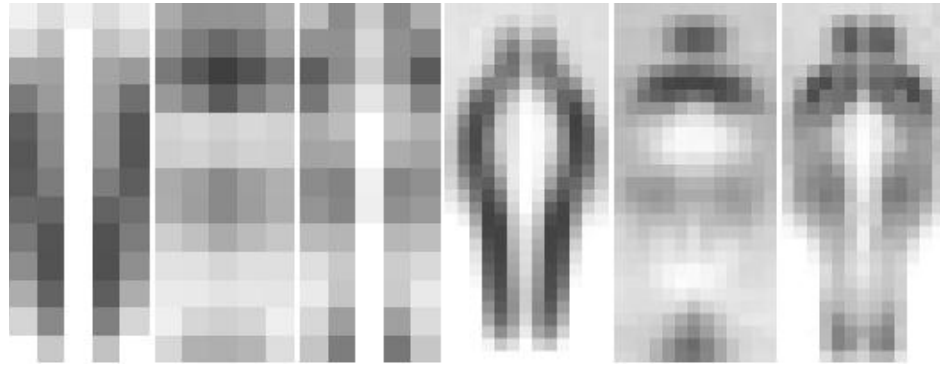


Figure 8: Biometric measurements for golden ratio [34]

The comparison of shape geometric parameters against reference values facilitates implementation, although these parameters are highly sensitive to slight variations in scale, orientation and position. Hence, abstract representations have been developed for robust feature representation.

Poggio et al. [36][37] proposed Haar wavelets to represent shapes. The shape of interest is subjected to a Haar wavelet transform to obtain a set of wavelet coefficients. Each wavelet coefficient describes the relationship between the average intensities of two neighboring regions. The transform is run using three types of Haar wavelets to obtain

coefficients for the vertical, horizontal and diagonal orientations. The wavelet transforms are also run at two scales to represent features at coarse and fine levels. The end result, which is shown in Figure 9, is a set of coefficients used to identify feature points in the object of interest. Results indicate that the full body of a pedestrian can be represented as a 29 dimensional vector.



**Figure 9: Wavelet coefficients at different scales [36].
The three leftmost images represent the vertical, horizontal and diagonal coefficients at a 32 x 32 scale. The remaining images are at a 16 x 16 scale**

The histogram of oriented gradients (HoG) approach, proposed by Dalal et al. [38], remains the most robust feature representation in computer vision [39]. The gradient of each pixel is calculated and quantized into angular bins. Local contrast variations due to illumination conditions are reduced by grouping pixels and normalizing the quantized gradients. These quantized gradients form the histogram of oriented gradients.

Abstract representations, such as Haar wavelets and HoGs, use a window-sliding technique to extract shapes in an image. The image is divided into overlapping or non-overlapping windows and the feature is calculated over each window. The window size is typically fixed at 64 x 128 [36] [38] for pedestrian detection purposes. Detection at various scales is achieved by resizing the image and re-computing the corresponding feature representations.

2.2.2.3. Decision-making algorithms

A decision-making algorithm determines whether the calculated features belong to a pedestrian. The prevailing approaches can be broadly categorized [40] into template matching, decision theory analysis and artificial neural networks.

Template matching determines whether the shape belongs to a pedestrian by matching with pre-defined templates. The major disadvantage of template-based algorithms is the computational intensity required to include all possible orientations, scales and positions of pedestrians. Gavrilu et al. [41] defined a hierarchical approach that compares the templates shown in Figure 10 on a coarse-to-fine scale to reduce computational expenses. Another drawback is the strong dependence of performance on reasonable contour detection [42], which can be hindered by clutter and poor contrast. Some systems simplify the classification process [34] by assuming that pedestrians walk upright, while some provide supplementary cues such as stereo information [43] to determine the scale.

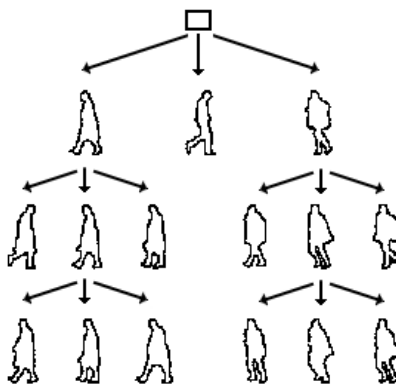


Figure 10: Template hierarchy proposed in [13]

Decision theoretic analysis processes the features extracted from a shape to form a feature vector. The approach employs statistical methods to classify the shape. Some

well-known decision theoretic methods include the minimum-distance classifier, the nearest-neighbor classifier and the minimum-mean-distance classifier.

Artificial neural networks are mathematical structures that simulate human information processing. The mathematical structure, which simulates human neurons, transforms itself based on the information flowing through it to minimize a cost variable. The transformation process, termed learning, accepts a series of examples and formulates the structure by identifying dominant relationships between the examples. The trained neural net is subsequently used to classify a test pattern. Artificial neural networks act as good classifiers by efficiently expressing non-linear decision surfaces. The huge variability in pedestrian shapes and poses demands an efficient non-linear classifier. A popular neural network-based classifier used in the decision making process is the support vector machine (SVM) [36] [38]. A detailed description of SVM is provided in Section 5.3.2.1.

The output provided by neural network-based classifiers is tightly coupled to the training procedure. Zhao et al. [17] used a back propagation procedure for training, where each neuron is assigned a weight based on the measured error rate for training examples. The process of boosting, proposed by Micilotta et al. [44], increases the accuracy of classifiers. The AdaBoost algorithm creates a highly accurate or strong classifier through a weighted combination of many inaccurate or weak classifiers. The weights are assigned in an iterative manner that minimizes the error rate. Initially, all classifiers are given equal weights. The algorithm then selects a weak classifier that will potentially minimize the error rate. The weights of training examples that are incorrectly identified by the classifier are increased. When the algorithm tests the remaining weak classifiers, it will

select a classifier that identifies those examples missed by the previous weak classifier. This technique results in an increased algorithm focus on the more difficult examples of the training set thereby leading to a minimum error rate. Cues such as motion and skin color [45] may be augmented with one of the above techniques to improve accuracy.

The overall complexity of training algorithms can be reduced by organizing the weak classifiers into a collection of cascaded layers [44]. A positive result from a simple first classifier triggers the evaluation of a second more complex classifier. A negative outcome at any point leads to the immediate rejection of the sub-window. A combination of cascade and AdaBoost algorithms therefore leads to a more efficient classification process.

Statistical shape models provide another solution to reduce computational time. Point distribution models (PDM) [46] generate a statistical model of the shape and variation of a human body from a set of training data. Each shape is represented by an n -dimensional vector as a set of n labeled landmark points. Given a new foreground image, an n -dimensional vector of the same landmark points is formulated by determining the translation, rotation and scale of the image. If the difference between the vector and the training data lies below a certain threshold, the shape is deemed to belong to a pedestrian.

2.3. Pedestrian tracking

One approach to tracking involves the comparison of certain geometric or visual features of a detected pedestrian in every frame. If a bounding box is drawn for each detected pedestrian, useful geometric features [45] include the area of the box, the vertical symmetry, the distance between box centroids, and the distance between median pixels and density. The visual features can be as simple as the average grey scale value or

average color value [47] of the region. The computation of a sum of squared differences metric (SSD) [48] for the color intensities of corresponding pixels in consequent frames accounts for color variability due to movements of the tracked object. Gonzales et al. [49] recommend using a weighted measure of geometric and visual cues for tracking purposes. The weights are assigned according to the distance of the pedestrian from the camera. Viola et al. [50] proposed a system that utilizes the walking pose information extracted through a cascade structure of Haar features and integral maps for tracking.

Tracking can also be stated as the problem of finding the position of a pixel p in the $(n+1)^{th}$ frame given the pixel position in the n^{th} frame. It is in this context that the optical flow parameter [51] is popular. Optical flow is defined as the velocity vector of each pixel in the image, estimated by evaluating the brightness gradient due to the apparent motion of the pixel. It has to be noted that optical flow refers to the apparent motion that is caused by the relative movement between the camera and the object. The movement in the z-direction in the real world is not be captured on a two dimensional frame.

The calculation of optical flow is based on the assumption that object pixels have the same brightness values over time. If $I(x,y,t)$ is the brightness value of a pixel at position (x,y) and time t , and it has moved by $\delta x, \delta y$ in a duration δt , the constraint equation is formulated as

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t)$$

Equation 1 : Constraint equation for optical flow

Applying a Taylor series expansion, the result is

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

Equation 2 : Modified constraint equation

where V_x and V_y are optical flow vectors for the pixel at position (x, y) .

The presence of two unknowns in a single equation requires additional constraints to arrive at a solution. The Lucas-Kanade algorithm [51] thus assumes that the position of a small pixel region between two consecutive frames remains more or less the same. Equation 2 can be extended to a pixel region by constructing a matrix, where each row in the matrix corresponds to a pixel. The solution for the constraint equation is obtained through the method of least squares, i.e. by minimizing the sum of the square of errors made by solving each equation in the matrix. The size of the pixel region must be chosen carefully. A large region allows background information to be dominant. A small region hinders identification of the tracked region.

The difference in optical flow for the different parts of a human body [52] makes it difficult to track pedestrians when a full body-based approach is used. Assuming that the optical flow for every pixel belonging to an object will be the same, it was initially proposed for background determination. However, the inability to detect static pedestrians discouraged its use in detection.

Statistical methods such as Kalman filtering, condensation algorithm, and mean shift algorithm are becoming popular for tracking purposes. Kalman filtering [53] models pedestrian movement as an equation with the current position, the previous position and time as parameters. Condensation algorithms [55] track randomly sampled pixels in a contour using a probabilistic approach. Mean shift algorithms [54] determine the position

of a pedestrian by minimizing the difference between a target histogram and the model histogram of image features of the detected pedestrian.

2.4. Miscellaneous techniques

Beardsley et al. [56] used stereo vision to detect the height and 3D shape of objects. This information, along with temporal analysis, is used to distinguish wheelchairs and pedestrians.

2.4.1. Zone-based detection

In a zone-based detection system, the field of view is divided into alerting, detection and tracking zones. When a significant portion of an object enters an alerting zone, it is extracted as the foreground. The extracted objects are tracked and classified while the object is in the detection zone. The pedestrian count is incremented whenever a large portion of the object clears the tracking zone.

A full-fledged algorithm to detect pedestrians in videos has been formulated by Sexton et al. [10]. The reference background frame is established by identifying 8 x 8 pixel blocks that remain stable for N frames. The background is refreshed every 3.3 seconds. The reference background is subtracted from subsequent frames and the resultant pixel values are compared against a threshold to determine the foreground. Incomplete object boundaries are closed using a binary closing operation. The foreground blob area provides a good estimate of the number of people in that region. The blob centroids with the smallest Euclidean distance in position for two consecutive frames are deemed to belong to the same object. Since the zones proposed by this system are horizontal, only right-left movements can be determined.

2.4.2. Vertical projection based methods

The vertical histogram projection [57] of a foreground component in a thresholded image is a plot of the number of white pixels versus a column index. If the projection is above a certain threshold, it is considered to belong to a pedestrian. This technique is based on the fact that a human body forms a peak in a projection of foreground pixels onto the X axis. When there are two or more people in a foreground blob, the distance between them causes a significant rise and fall in the vertical projection histogram as shown in Figure 11. Pedestrians are therefore counted by searching for relevant peaks and troughs in the projection histogram. The shadow pixels, which are much lower in number in the projection histogram, are suppressed. A maximum-likelihood matching algorithm is used to determine if an object is a pedestrian.

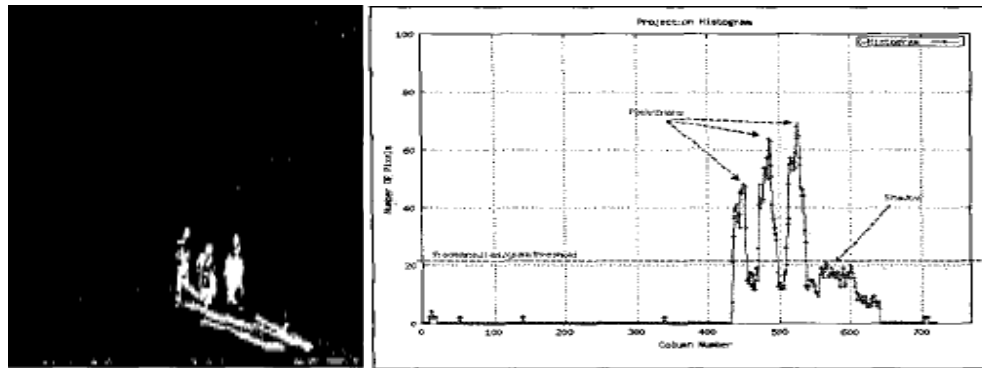


Figure 11: A thresholded image and its vertical projection histogram [57]

2.5. Summary

The highlights of the discussed techniques are presented in

Table 2. This thesis has employed HoG as a pedestrian detection technique, based on a study conducted by Caltech [39].

Table 2: Summary of different pedestrian detection techniques

Technique	Candidate Generation	Advantages	Disadvantages
Zone based detection	Computer Vision	<ul style="list-style-type: none"> • Less calculations • Sufficient accuracy • Large coverage area 	<ul style="list-style-type: none"> • Only pedestrians parallel to the image plane are detected • Hardware limitations
Histogram of Oriented Gradient	Computer Vision	<ul style="list-style-type: none"> • High accuracy 	<ul style="list-style-type: none"> • Difficulty to detect pedestrians in various poses
Hough Transform	Computer Vision	<ul style="list-style-type: none"> • Ability to detect at various scales 	<ul style="list-style-type: none"> • Computationally complex • Only pedestrians parallel to the image plane are detected
Distance Transform	Computer Vision	<ul style="list-style-type: none"> • Less computational complexity 	<ul style="list-style-type: none"> • Inability to detect objects close to camera
Haar Wavelet	Computer Vision	<ul style="list-style-type: none"> • Considers various poses and scales 	<ul style="list-style-type: none"> • Computationally complex • Low accuracy 81.5%
Stereo Vision	Computer Vision	<ul style="list-style-type: none"> • High accuracy • Real-time detection • Robust against lighting changes 	<ul style="list-style-type: none"> • Inability to detect people at the same depth as background • Limited coverage area
Laser Scanner	Time-of-flight	<ul style="list-style-type: none"> • High accuracy • Easy setup 	<ul style="list-style-type: none"> • Computationally complex • Not robust against weather.
Passive Infrared sensor	Heat generated by human body	<ul style="list-style-type: none"> • Economical method • Robust against lighting changes 	<ul style="list-style-type: none"> • Dependence on clothing • Limited coverage area • Inability to detect still pedestrians
Microwave RADAR	Doppler Effect	<ul style="list-style-type: none"> • Economical method 	<ul style="list-style-type: none"> • Unsuitable for counting
Ultrasonic sensor	Doppler Effect	<ul style="list-style-type: none"> • Economical method 	<ul style="list-style-type: none"> • Unsuitable for counting
Active Infrared sensor	Doppler Effect	<ul style="list-style-type: none"> • Economical method 	<ul style="list-style-type: none"> • Unsuitable for counting

CHAPTER 3

AUTOSCOPE SOLO TERRA

This chapter describes the capabilities of the Autoscope Solo Terra, which is used to count pedestrians. The Solo Terra is typically used to monitor the traffic flow of vehicles. The camera can be configured to calculate traffic data such as volume and speed.

3.1. Solo Terra camera-based system

The Solo Terra system consists of the following components:

- Solo Terra, the camera that collects data.
- Computer hardware, a 32-bit Windows Operating System based PC that aids in configuring Solo Terra.
- Terra Interface Panel (TIP) that interfaces communication between the Solo Terra and computer hardware.
- Computer interface software (Autoscope Software Suite) that configures the Solo Terra for traffic data collection.
- An Ethernet cable to connect the TIP and computer hardware.
- A power source capable of providing power to the camera and TIP.
- A mounting structure that is used to mount the camera at a specific height and angle.

A prototype of the setup is illustrated in Figure 12.



Figure 12: The Solo Terra camera based system

3.1.1. Autoscope Solo Terra

The Autoscope Solo Terra includes an integrated Machine Vision Processor (MVP) consisting of a camera and a TI DaVinci TMS320DM6446 dual core processor [2][58]. The camera streams video to a PC over a 3-wire power line. The TMS320CC64x+ core in the processor performs sophisticated digital signal processing operations. General purpose processing is handled by the ARM926 core. Multi-threaded software performs real-time operations to extract traffic data and transmit detector outputs while simultaneously streaming MPEG-4 video [2][58].

3.1.2. Autoscope Software Suite

The Autoscope Software Suite provides an interface on the attached PC to configure the Solo Terra. It consists of a GUI interface that allows users to select detection zones and retrieve traffic statistics collected by the camera.

3.1.3. Terra Interface Panel

The Terra Interface Panel or TIP acts as an interface for communication between Solo Terra and a PC or a traffic controller. The TIP supports “3-wire” connection for up

to eight Solo Terra sensors, a traffic controller and an Ethernet connection to a PC. The basic capabilities of TIP are illustrated in Figure 13.

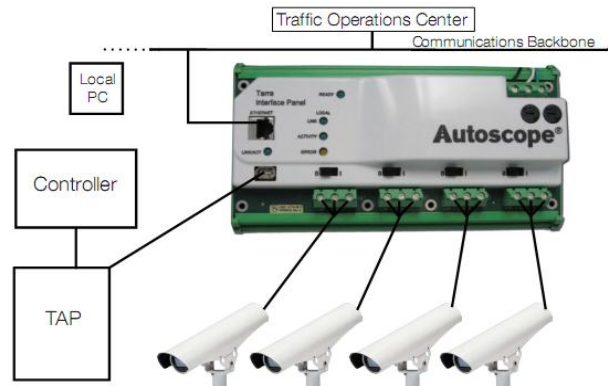


Figure 13: Terra Interface Panel [59]

The TIP can support throughputs of up to 6Mbps, which is sufficient for transmitting video and traffic data [60]. The system reliably delivers data packets by adjusting transmission rate based on a varying signal-to-noise ratio. In addition to serving as a communication interface, the TIP also protects Solo Terra sensors from cable transients and surges.

3.2. Components of the Autoscope Software Suite

The constituent software of the Autoscope Software Suite can be accessed through a client application called the Autoscope Network Browser. Upon user prompting, the Network Browser scans the Ethernet channel and detects Autoscope products present on the network. The user can then control each of the detected devices. The relevant software functionality interfaces provided by the software suite are described below.

3.2.1. Autoscope Configuration Wizard

The Configuration Wizard allows users to create detection zones and configure the Solo Terra mode of operation. The detection zones are described in Section 3.3.1. The

Solo Terra can operate in intersection mode or highway mode. Intersection mode requires additional hardware.

3.2.2. Autoscope Detector Editor

The Detector Editor allows users to modify, download and upload detector configurations to an Autoscope device. The detector software monitors and collects traffic statistics from user-defined regions in the image. A detector configuration file contains information such as the locations that are being monitored, the detector parameters and the detector size along with calibration data and video format. Only one configuration file can be run by a Solo Terra at a time. The current image captured by a Solo Terra may be uploaded to determine the locations for detector placements. The image size is set to 352×240 pixels for NTSC format and 352×288 pixels for PAL format. The detector facilities supported by the camera are described in Section 3.3.2.

3.2.3. Autoscope Data Collector

The data collector allows users to transfer the traffic data present on the 12.5MB flash memory card of the camera to a PC. The data stored on flash gets overwritten after utilization of its maximum capacity. The frequency at which data is polled and stored on the flash is controlled by the user at the time of configuration. The size of an individual poll record for each detector varies according to the data it collects. Thus, the duration for which the camera is capable of storing data without getting overwritten depends not only on the flash capacity and the polling interval but also on the type and number of detectors used. The Solo Terra supports temporary storage of polled data on RAM present in the camera. The detection session data are stored in text-file format with a timestamp on the

PC. The rate of polled data retrieval from the Solo Terra can be as low as one second and can be incremented in steps of a second up to any number of days.

3.2.4. Autoscope Video Player

The Video Player allows video streaming and detector activity monitoring. The frame rate of streamed video depends on the number of detectors in the configuration, the video compression algorithm and the link speed. The video can be streamed in uncompressed, JPEG or MPEG4 format. The Solo Terra is capable of recording videos in a standard format when a detector configuration is not loaded into the camera.

3.2.5. Autoscope Property Editor

The property editor aids in setting preferences for video quality, streaming frame rate, and time zone of operation. An incorrectly set time zone leads to operational errors. The Solo Terra can infer the time zone from user-defined latitude and longitude values. The software allows for the creation of user-logins to avoid the misuse of configuration files.

3.2.6. Autoscope Software Installer

The installation of the Autoscope Software Suite on the Solo Terra is mandatory for downloading configuration files. The installer interface provides a means to download and install the latest versions of the Software Suite on the camera.

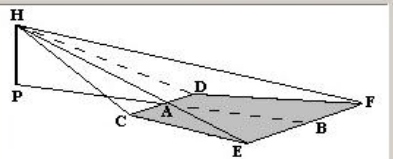
3.2.7. Autoscope Field of View Calculator

The field of view is defined as the region captured on the camera sensor. The focal length of the lens defines the relationship between the field of view and the distance from the back of the lens to the target object. The software can calculate the field of view based on user-defined parameters. A screenshot of the calculator is shown in Figure 14.

Autoscope Field of View Calculator (COPYRIGHT 2002 Image Sensing Systems)

Enter all measurements using the same unit system (i.e. Use feet or meters, but do not mix them).

Camera CCD Size	1/2 inch - 2004
Focal Length (mm)	8.0 - Solo, 2004
PH = Camera Height	40
PA = Distance to Near FOV	100
PB = Distance to Far FOV	200
CD = Near Field of View Width	50
EF = Far Field of View Width	100
Maximum Focal Length (mm)	13.8
CHD = EHF = Horiz. Angle of View	26.1
CHE = DHF = Vert. Angle of View	19.8
PHB = Vert. Angle to Top of View	88



H = Camera Location CDEF = Field of View on Ground

Camera Rotation
☒ None ☐ 90 Degree

Calculate Help Close

Top Edge of Field of View Above Horizon!

ENTER: PA, PH, CD
 ENTER: PA, PH, PB
 ENTER: CHD
 ENTER: CHE
 ENTER: PA, PH, Focal Length
 ENTER: PA, PH, CHD
 ENTER: PH, Focal Length, CD
 ENTER: PB, PH, CHE
 ENTER: CD, CHD, PH
 ENTER: PA, PH, CHE
 ENTER: PA, CD, Focal Length
 ENTER: PA, CD, CHD
 ENTER: Focal Length

Figure 14: Screenshot of the Autoscope Field of View calculator

3.3. Detection capabilities

The Solo Terra camera is capable of estimating the traffic data summarized in

Table 3 by evaluating monitored regions. Data collection continues without any human intervention after camera mounting and configuration at a suitable height and angle. Various Solo Terra configuration options are available to allow for interfacing with traffic controllers. The detection capabilities of the Autoscope Software Suite have been optimized for vehicular traffic monitoring.

Table 3: Traffic data collected by the Solo Terra camera [61]

Parameter	Definition
Average flow rate	Average flow rate, in vehicles counted per hour.
Total volume count	Total number of vehicles that have passed through the selected detector during the time interval.
Arithmetic mean speed	Average speed of all vehicles passing through a specified speed detector during the polling interval (reported in km/h or mi/h).
Vehicle class count (A – E)	Number of vehicles classified by five user-defined length intervals
Average time headway	Average number of seconds from leading edge of a vehicle to leading edge of the following vehicle.
Average time occupancy	Percentage of time that vehicles occupy a detector in a measurement interval
Level of service	Rates a roadway's traffic capacity from optimum ("A") to ineffective ("F"). Each roadway is capable of handling different levels of traffic, measured by speed or flow/capacity.
Space mean speed	Average speed of all vehicles occupying a given section of highway over some specified time period. SMS is computed by dividing distance with an average travel time
Space occupancy	Percentage space on a stretch of roadway, typically 1Km that is occupied by vehicles at a given point in time.
Density	Number of vehicles per lane traveling over a unit length of highway at an instant in time.
Interval occupancy	Each occupancy interval is a maximal time interval during which one or more vehicles are in the region of interest.
Interval net time gap	Sum of the individual net time gaps of the detector over the same interval.
Net time gap	Net time gap from the previous vehicle to the detected vehicle, in milliseconds
Vehicle length	Length of vehicle
Vehicle distance	Inter-vehicle distance between two vehicles on a roadway

The Solo Terra allows for the creation of detection zones for vehicle monitoring and detectors, for object detection. Detection accuracy is robust against brightness variations since changes in pixel values due to movements of the sun or light sources inside tunnels are considered.

3.3.1. Detection zone

A detection zone is a region where a detector can be placed. A detection zone can be sized by the user to accommodate either a single lane or multiple adjacent lanes in a freeway or an intersection. Each zone is associated with a speed detector, an incident alarm, a speed alarm and a scheduler. The zones are non-overlapping and contiguous since they monitor vehicles in freeway lanes or in intersections. Stop zones can also be configured to alert an operator if a vehicle stops in the monitored region. Detectors for vehicular traffic detection can be placed in a single lane or in multiple lanes associated with a detection zone.

3.3.2. Detectors

Traffic data are collected in image regions that are covered by detectors implemented as geometric shapes. Detectors appear as line, box, or arrow shapes overlaid on an image region. The maximum number of visible Solo Terra detectors has been set to 99. The detection speed is affected by the number of detectors in an image. Detector regions can overlap one another if desired. A detector is turned ON whenever the pixel values in that region differ from the background. The data is updated whenever a detector undergoes transitions. The Solo Terra offers a wide range of detectors to calculate the traffic parameters listed in Table 4. These detectors are described in detail in the following sections.

Table 4: Type and functionalities of available detectors in Solo Terra [61]

Type	Functionality
Count	Counts vehicles passing through the detector region. The default dimensions of detector should fall in the following range: Length – 10-200 pixels or 2-100 feet Width – 2-100 pixels or 1-100 feet
Speed	Calculates the speed of vehicle and measures the vehicle length.
Presence	Reports the presence or absence of an object. The default dimensions of detector should fall in the following range: Length – 10-300 pixels or 1-200 feet Width – 2-150 pixels or 1-200 feet
Boolean	Combines output of detectors to obtain a customized output
Incident	Reports possibility of traffic congestion. Used in conjunction with a scheduler to avoid false alarms.
Scheduler	Schedules the time and duration at which a detector should function by listing rush-hours to reduce false alarms.
Speed Alarm	Rings an alarm if speed exceeds a certain value. The speed can be that of individual vehicles or averaged value.
Smoke	Detects smoke in an area
Stop Line	Reports whether a vehicle has stopped outside the designated stop lines on the road. Does not update any traffic parameters.
Pedestrian and Debris	Detects pedestrians and debris in a tunnel.
Outdoor Lane	Detects stopped, slow or wrong-way vehicle in outdoor conditions and under-bridge decks. Requires good quality color video input signal.
Tunnel Lane	Detects stopped, slow or wrong-way vehicle in tunnel applications under controlled lighting conditions where traffic may approach or recede in a single lane.

3.3.2.1.Deciding locations for detector placement

The following factors need to be considered while determining locations for detector placement.

- Uniform background - Placing detectors over high-contrast regions such as stop lines, lane markers and crosswalks may trigger false alarms. For optimal results, detectors have to be placed over sections which have a uniform appearance. The false

triggering of detectors placed in high-contrast regions due to slight camera movements is illustrated in Figure 15.



Figure 15: Anomaly due to non-uniform background

- Movement and rapid motion - Rapid movements such as those of wires and tree branches may trigger false alarms.
- Occlusion - Occlusion should be avoided by increasing the camera mounting height and by adjusting the camera position according to the field of view.
- Distance of the camera from the monitored area - At long range of greater than 70 meters, the jagged appearance of a slanting detector, as shown in Figure 16 affects the detection accuracy by triggering false alarms.



Figure 16: Jagged detector - The detector appears like a stair of steps

3.3.2.2. Detector sizing

The size of a detector is dependent on its location in the image and the size of vehicles that have to be identified. Detectors that are located farther from the Solo Terra should be sized smaller than those closer to the camera since vehicles appear smaller at long range. Even though the detectors are initially created using default size settings, they can be resized to cover any amount of area in the image. Specific sizing recommendations exist for each detector type. For example, the recommended sizing for a count detector is 0.6 to 1.2 meters (2 to 4 feet) in thickness and 1.5 to 2 car widths in length. Speed detectors are automatically sized by the software depending on the calibrated data entered at the time of configuration.

In general, extremely small detectors have a high false alarm rate because of the sensitivity towards shadows or camera motion. Large detectors may have a higher rate of missed detections since the necessary details to make a proper detection decision will be averaged out [61]. For example, a count may only be incremented once when two vehicles pass through a detector region at the same time.

3.3.2.3. Properties of detectors

Important detector parameters include:

- Traffic direction
- Background refresh rate - This parameter is the guaranteed minimum time an object must remain stationary before it is considered as background. The values range from 20 seconds to 600 seconds. For freeway applications, a background refresh rate of 30-60 seconds is recommended.

- Night reflection – This value indicates the compensation for roadway reflections that cause false detections. If this parameter was not used, light projected from a vehicle's headlights may activate a detector and the vehicle length would be measured as the combined length of the beam and the vehicle.
- Crosslane or downlane orientation – A placement that is parallel to the lane is indicated by specifying downlane orientation. A placement that is perpendicular to the lane is specified by a crosslane orientation.
- Shadow processing -This parameter prevents false detections caused due to shadows cast by vehicles in adjacent lanes. The processing is normally turned OFF at midday by the software because shadows are mostly absent at that time. Exceptions, such as small fast clouds, require shadow processing to be turned ON during midday. Midday is defined as the period when the sun is higher than N degrees, where N is based on the latitude/longitude settings for the detector file. The lighting conditions and the direction of the source with respect to the detector are specified in order to effectively compensate for the movement of the sun or light sources inside a tunnel.

3.3.2.4. Presence Detector

A presence detector reports the presence or absence of an object in the region of interest. Presence detectors appear as straight lines, as shown in Figure 17, that are typically placed parallel to the flow of traffic. The parameters associated with a presence detector are background refresh rate, shadow processing and traffic direction. The detector can be programmed to consider the presence of vehicles moving in a particular direction.



Figure 17: Illustration of presence detectors.
The directional detectors have been highlighted in red.

3.3.2.5. Count Detectors

A count detector counts vehicles passing through the detector region. The count detector can be used to calculate speed, traffic volume and occupancy statistics since they are optimized to distinguish between closely grouped vehicles. Even though there are no restrictions on the orientation of a detector, they are generally placed perpendicular to the direction of motion for accurate counting. The detectors appear as straight lines in an image as shown in Figure 18. Count detectors are also associated with a background refresh rate, traffic direction and shadow processing.



Figure 18: Count detector as a series of white lines that cross the intersection [61]

3.3.2.6.Speed Detector

Speed detectors measure the speed and length of a vehicle and classify them into classes A through E based on the measured length. A speed detector, which appears as a rectangular region in the image, is associated with a count detector as shown in Figure 19. The length of the rectangle is fixed by the software depending on the minimum and maximum speeds that can be reported by the detector. The limits on the reported speed imply that values outside the stipulated range shall be clipped. The width of the rectangle is controlled by the dimensions of the associated count detector.

Traffic data are only processed after the count detector changes state from ON to OFF. After the count detector turns OFF, the length of the vehicle traversing the rectangular zone is measured along with the time taken to traverse the zone. Speed is calculated as the ratio of the length of the rectangle to the time taken for traversal. Since data processing only starts after the count detector turns OFF, the results are “unknown” if a vehicle approaches the detector from the opposite direction i.e. enters the rectangular region first and then activates the count detector. Hence, for accurate results, a vehicle must approach the camera.

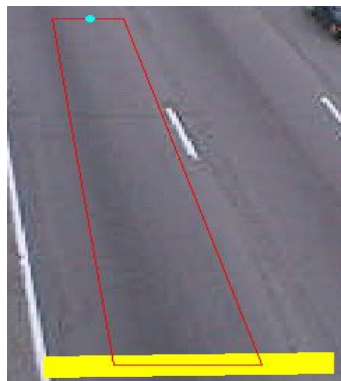


Figure 19: Illustration of a speed detector.
The red rectangle represents the speed detector and the yellow region is the associated count detector

Calibration improves the accuracy of speed detectors by compensating for traffic traveling at different road surface heights. Higher elevation traffic in the field of view appears to move faster than lower elevation traffic traveling at the same speed [61]. The measured speed is multiplied by an adjustment factor to obtain a new reported speed. A minimum vehicle length is specified to guarantee a minimum ON time for the detector. Otherwise, at night, vehicle headlights would activate the detector, resulting in a short vehicle length measurement.

3.3.2.7. Boolean Detector Function

A Boolean detector function combines the outputs from two or more detectors into a single customized output. A Boolean logic function states the conditions that must be satisfied to generate an ON value for the customized output. Table 5 lists the Boolean operations that are supported by the software.

Table 5: Boolean operations supported by Autoscope Software Suite [61]

Operation	Conditions for the final output to be ON
OR	At least one of the member detectors turns ON.
AND	All the member detectors must be turned ON.
NAND	At least one of the member detectors turns OFF.
NOR	All the member detectors must be turned OFF.
M of N	At least M members out of N member detectors must be ON.

An example of the usage of Boolean detector function is shown in Figure 20 where two directional presence detectors are ORed to report wrong-way vehicles.



Figure 20: ORing of two directional presence detectors

3.3.2.8. Local Contrast Detector

A local contrast detector assists in monitoring the quality of images that are processed by the camera. The operator is notified whenever the contrast in the detector area is insufficient due to dirty faceplate conditions caused by snow or ice. The detector appears as a rectangular box with a central reference line, as indicated in Figure 21. The reference centerline must be centered on and oriented parallel to a high-contrast boundary in the image. The local contrast detectors are typically placed near stop-lines at intersections.



Figure 21: Local contrast detectors

They are placed on regions where contrast variations may occur due to snow

3.3.2.9. Detector stations and label detectors

Detector stations and label detectors ease the collection of the traffic data listed in

Table 3. Detector stations collect and report data gathered over specific time intervals. For example, if a time interval is fixed at ten minutes, the detector reports traffic parameters for ten minutes and resets the parameter values to gather data for the next time interval. Detector stations connect to count, presence, incident, detector functions, speed, tunnel lane, outdoor lane, stop line and label detectors. They appear as fixed size, tiny squares and can be placed anywhere on an image. A detector station can also connect to a label detector that is specifically designed to show polling data and the names of the detectors on the streamed video. Such a scenario is illustrated in Figure 22.

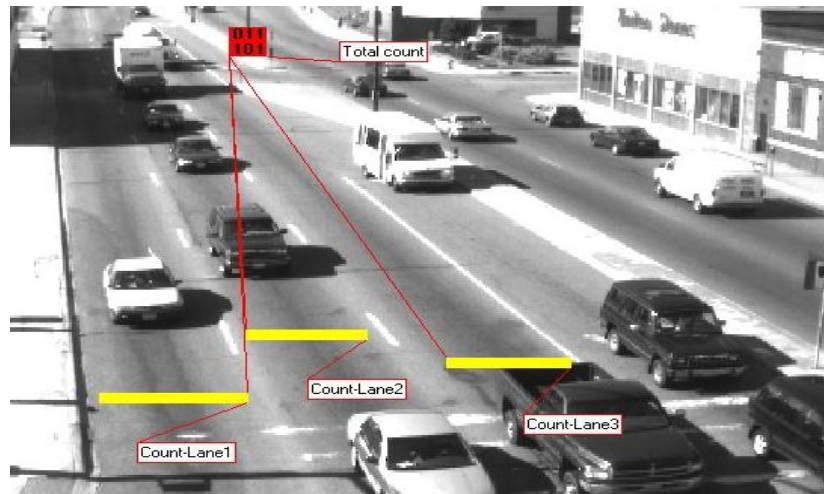


Figure 22: Detector Stations and Label detectors.
The red box is a detector station connected to the three count detectors shown in yellow. All the count detectors and the detector station are named using label detectors.

3.3.2.10. Pedestrian and Debris detectors

Pedestrian and debris detectors report the presence of pedestrians or debris in a frame. They are not considered for counting due to the following reasons.

- Multiple cameras are required for the accurate detection of pedestrians.
- They have high sensitivity to lighting conditions and camera position.

- Multiple pedestrian detectors cannot be placed on the image. - Only one detector can be configured to cover the entire frame. Areas to be excluded must be marked. A single ON signal is generated irrespective of the number of pedestrians in the image.
- The software is only available on request from an Autoscope specialist.
- The inability to connect to a detector station implies that a count cannot be estimated from the pedestrian detectors.

Count detectors can connect to a detector station if users need to obtain the count directly from the Solo Terra. The flexibility of placing multiple count detectors over an image along with the possibility of combining their results using Boolean functions makes them a good choice over the pedestrian detector option available in the Solo Terra. For pedestrian detection and counting, count detectors with detector stations and label detectors need to be properly configured.

CHAPTER 4

SYSTEM DESIGN USING THE AUTOSCOPE SOLO TERRA

This chapter provides implementation details of the Autoscope Solo Terra-based system. A robust camera-mounting structure was constructed to support field experiments. A software which increments count in real-time based on the information transmitted by the camera was developed for counting purposes.

4.1. Design of the hardware support structure for the Solo Terra

The hardware support structure for the Solo Terra was built at the Mechanical Engineering workshop at the University of Massachusetts, Amherst. The structure was designed by Cheng Zhang of Civil Engineering Department to satisfy the following requirements.

- Provide a mechanism to adjust the camera height and angle. The maximum height supported by the structure is 15 feet.
- Low cost (e. g. hundreds of dollars)
- Ease of transportation
- Ready availability of raw materials
- Deployment on uneven, sloped surfaces

The support structure shown in Figure 23 employs an eight foot stepladder as a stable base for mounting the camera. The lightweight ladder is easy to transport. A hole drilled in the top of the ladder accommodates a pipe which acts as the main mounting pole. This pipe is fixed to the ladder with the help of angle iron and C-shaped clamps. Short, sharpened reinforcing bars (rebars) on the bottom end of the pipe penetrate into the ground thus fixing the position of the structure. Four additional pieces of rebar may be

added for further support by means of C-shape clamps. The position of the rebars on the ladder can be adjusted to ensure penetration into uneven ground.

A smaller pipe inserted into the top end of the main pipe facilitates camera height adjustment. The length of the pipe can be varied from 2 ft to 4 ft. Hence, the total length of the mounting pole ranges between 12 ft and 14 ft which satisfies the system requirements. A flange was installed on the top of the pipe to connect to the adjustable camera bracket. If the length of the rebar and the camera bracket is taken into account, the maximum height at which the camera can be mounted is 15 ft.



Figure 23: Mounting structure along with the raw materials

The structure provides access to the adjustable part of the pipe and the camera. This access facilitates the adjustment of the height and angle of tilt of the camera by allowing an operator to climb up the ladder. The angle of tilt can be varied using a wrench. The camera can be manually swiveled and inclined by the adjustable bracket

attached to it. All individual parts can be fully assembled on the field by two persons in less than twenty minutes.

An insulating enclosure has been provided for the interface panel to ensure safe operation. The box is placed beside the step ladder. A long power cable connects the TIP to a power source. The interface panel supplies the necessary power to the camera through another power cord.

4.2. Software resources used for pedestrian detection

The primary objective behind the implementation of the Autoscope Solo Terra-based system is to determine whether existing vehicle detection mechanisms and equipment are suitable for pedestrian detection. Hence, the Autoscope Software Suite, which supports basic detection functionalities, was selected as the interface software for the PC. The more sophisticated and high-priced Software Development Kit (SDK) available from Autoscope was found to be unneeded.

4.3. Detector configuration for the Solo Terra

Detectors identify object movement in certain image regions captured by the camera. The monitored regions are selected by the user using the Detector Editor software described in Section 3.2.2. The software configuration requirements needed to detect pedestrians are listed below.

- The latitude and longitude settings entered through the Configuration Wizard are set for the desired test location.
- The camera is mounted at a height and angle which includes the entire width of the sidewalk in the field of view.
- Count detectors, detector stations and label detectors are employed for counting.

- The locations of detectors ensure that all pedestrians in the frame are counted only once.
- A sufficient number of count detectors are placed to achieve 85% accuracy.
- Shadow processing is enabled to avoid false detections.
- The background refresh rate is fixed to avoid the classification of slow pedestrians as background.

It has been determined through experiments that the detector facilities supported by Solo Terra cannot distinguish between objects. Each detector region is in one of two states – ON or OFF as described in Section 3.3.2. A ON state occurs whenever the current pixel values in the monitored regions of the captured video are different from the background pixel values, as illustrated in Figure 24. Background pixels are estimated by capturing snapshots of the field of view at intervals determined by the background refresh rate. Ideally, a sidewalk without pedestrians constitutes the background.

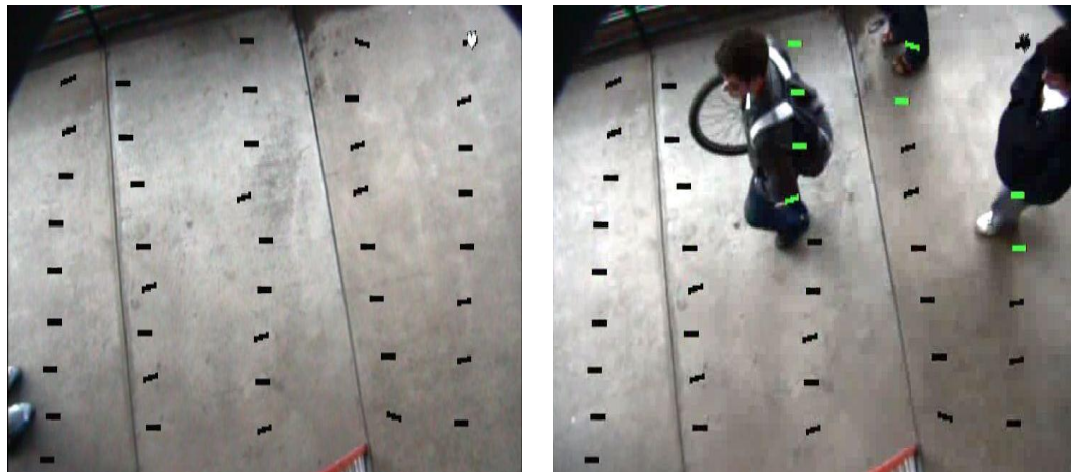


Figure 24: Activation of detectors by pedestrians
 Left to right - Detectors remaining OFF (black rectangles) in the absence of pedestrian traffic; Detectors turning ON (green rectangles) when pedestrians cross the region.

The OFF-ON transitions signify the entry of an object into the detector region while an ON-OFF transition signifies an exit. It is necessary to ensure an OFF-ON-OFF transition for every pedestrian crossing the detector region since counts are updated based on these transitions. This requirement translates to capturing the “gap” between pedestrians by suitably mounting the camera and appropriately sizing the detectors.

If the camera directly faces pedestrians it will fail to capture the “gap” between pedestrians who walk behind one another. Such an arrangement causes frequent undercounts and misses which lead to a high error rate. During field experiments it was found that pedestrians walk behind one another more frequently than they walk beside each other. Hence, the camera must be placed on the side of walkway rather than facing pedestrians.

Detector configurations in vehicle monitoring applications consist of one detector per lane, each of a length that covers a single vehicle but does not stretch beyond the ends of the lane. The statistics collected by this type of configuration yields accurate results for vehicle monitoring since vehicles generally follow traffic lanes i.e. a vehicle can occupy only one lane at a time except while changing lanes. Unlike vehicles, pedestrians are of different sizes and do not walk in strict lanes on sidewalks. Thus, the use of multiple similarly-sized detectors of the average size of a pedestrian’s shoulders may not provide the required count accuracy. Undercounts and misses may occur due to occlusion. Overcounts may occur since a pedestrian may cover multiple detectors. Such a scenario is illustrated in Figure 25. This issue motivates the use of multiple small detectors.



Figure 25: Multiple detectors turning ON due to irregular movements of pedestrians

An efficient method of counting pedestrians using the Solo Terra was developed with the help of Chaoqun (Enzo) Chia from the UMass, Amherst Civil Engineering Department. A configuration of two columns of closely-spaced, uniformly-distributed count detectors of size 9×2 pixels was used, as shown in Figure 27. The small spacing and size of the detectors ensure that for a mounting height of greater than 9 feet, the number of detectors triggered ON by a single pedestrian can be approximated to a be constant. This constant was determined during field experiments. Detectors are placed near the center of the image to ensure that pedestrians walking in opposite directions are included in the statistics. The principle of redundancy also reduces the overcounts caused by slight movements of the camera and the shadows of moving tree branches.

4.4. Pedestrian counting algorithms using the Solo Terra

The detector state transitions are recorded and dumped into a text file on a PC at regular intervals through the Terra Interface Panel interface. The Solo Terra was configured to collect the states of count detectors and transfer them to the PC once every second. The retrieval rate of one second ensures real-time count updating. The text file

contains information such as the detector number, date, time, ticks and state as shown in Figure 26.

```

detector;   date   ;    time   ; ticks ;state
number
110; 3/25/2011;11:58:47 AM;1470216;0;
112; 3/25/2011;11:58:47 AM;1470316;0;
111; 3/25/2011;11:58:47 AM;1470383;0;
112; 3/25/2011;11:58:49 AM;1473119;1;
112; 3/25/2011;11:58:50 AM;1473453;0;
112; 3/25/2011;11:58:51 AM;1474754;1;
112; 3/25/2011;11:58:51 AM;1474888;0;
118; 3/25/2011;11:58:52 AM;1475789;1;
117; 3/25/2011;11:58:52 AM;1475855;1;
118; 3/25/2011;11:58:52 AM;1475956;0;
117; 3/25/2011;11:58:52 AM;1476022;0;

```

Figure 26: Example of polled data information

Detector numbers are used to identify detector regions. The state field indicates whether the detector is currently OFF or ON. The ON state is denoted by a '1' and the OFF state is denoted by a '0'. The tick field is a 32-bit integer that increments every millisecond. Thus, depending on the tick value, the duration for which a detector stays in a particular state can be determined to millisecond granularity.

Software was developed which performs operations on the text file to determine pedestrian counts. Three approaches were implemented to count pedestrians. These approaches are explained in the next three subsections.

4.4.1. State averaging algorithm

The state averaging approach uses a configuration of two columns of 15 detectors each, as indicated by the red rectangular region in Figure 27. The number of OFF-ON transitions are counted and divided by a constant. The required constant varies according to the mounting height and angle of the camera. The time in milliseconds and the position coordinates of detectors are not considered, although during experiments it was observed that vertically adjacent detectors are most likely to have ON state at the same time.



Figure 27: Detector configuration used in the state averaging approach

4.4.2. State matrix approach

The state matrix approach uses a configuration of a single column of 15 detectors as illustrated in Figure 28. Unlike the state averaging approach, the exact transition time and locations of detectors are noted. The positions of the farthest and the closest detector are observed to estimate the area of movement. Based on the observation that the number of detectors triggered ON by a pedestrian for a specific camera mounting height is constant, the pedestrian count is incremented once, twice or thrice depending on the position of the closest and the farthest detectors in the ON state at a given time.



Figure 28: Detector configuration used for the state matrix approach

4.4.3. M of N approach

The M of N approach uses a single column of detectors, as shown in Figure 29. The approach uses the Boolean functions embedded within the camera. Grouped detectors are overlapped such that a pedestrian covers a majority of detectors in a single group. If there are N detectors in each group and at least M detectors in a group turn ON simultaneously, the count is incremented. The count increment depends on the number of groups which are activated.



Figure 29: Detector configuration for the M of N approach

CHAPTER 5

VISION BASED SHAPE DETECTION

5.1. Overview

A computer-vision based shape detection technique was implemented as second pedestrian detection approach. As described in Chapter 3, potential pedestrian candidates can be extracted from image frames through background subtraction. A pre-determined number of frames are buffered at regular intervals to determine the background. The background is subtracted from subsequent frames to determine the presence of new objects in the frame. This action is followed by classification that identifies the pedestrians in the foreground. The pedestrian count is updated whenever a pedestrian disappears from the field of view.

The classification of shapes into pedestrian and non-pedestrian categories is carried out by scanning the frame for a “Ω” shape formed by the head and shoulders of a pedestrian [12] whenever pixel values are distinguished from the background. Shapes are represented by means of a histogram of oriented gradients (HoG) [38]. As discussed in Chapter 3, the decision making algorithm is implemented as a support vector machine (SVM) [62]. The output of the SVM is 1 if the algorithm recognizes an “Ω” shape and 0 otherwise. This second implemented pedestrian detection and counting process is summarized in the flowchart in Figure 30.

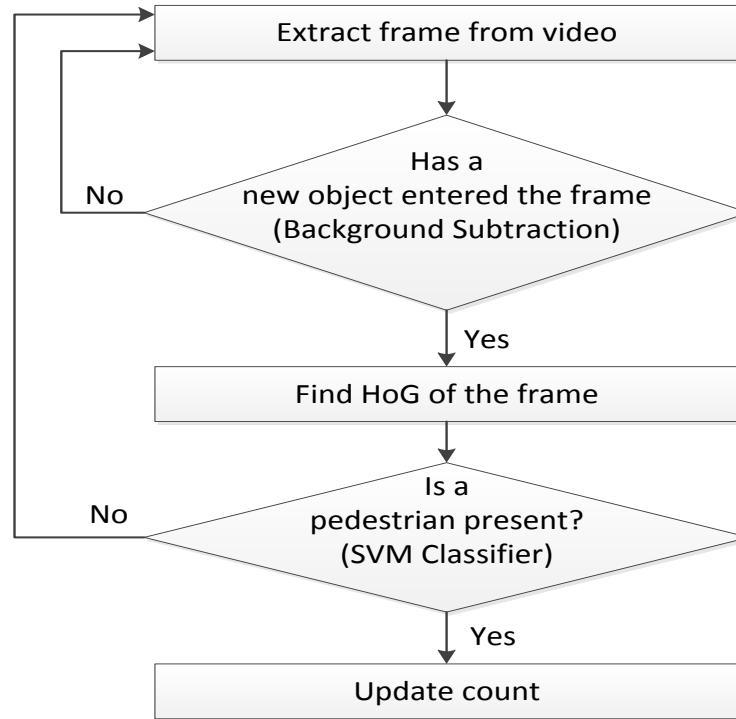


Figure 30: Procedure followed for the automated detection and counting of pedestrians based on histograms of gradients (HoGs)

5.2. Background determination and subtraction

The purpose of background subtraction is to identify whether new objects have entered the frame that is being processed. Frames are buffered to determine the background. The estimated background is subtracted from every frame to identify new objects and the difference is thresholded to reduce the effects of slight brightness variations in pixels. Hence, the performance of any background subtraction algorithm is affected by the determined background as well as the contrast of foreground objects against the background.

A reasonable compromise between computational complexity and performance is achieved by employing the approximate median algorithm proposed by McFarlane et al. [63] to determine the background. The approximate median method considers the median

of N buffered frames as the background. This calculated background is subtracted from subsequent frames to extract the foreground objects.

The background is determined iteratively by converting the buffered frames to grayscale images and finding the median values of pixels at each location. The background is initialized to the first buffered frame and the following frames are processed sequentially. If a pixel in the current frame has a value that is larger than the corresponding pixel in the background frame, the value of the background pixel is incremented by one. If the value of the current pixel is less than the value of the background pixel, the background is decremented by one. Eventually, the pixels converge to the median where half the input pixel values are greater than the background, and half the values are less than the background if there is no movement in the scene. The time taken to converge to the median value depends on the number of buffered frames, the frame rate and the amount of movement in the scene. The background determination process is summarized in Figure 31.

The number of buffered frames required to determine the background depends on the activity in the captured region and the frame rate. The background can be periodically updated to accurately identify new foreground objects. The updated background is subtracted from subsequent frames to determine new frame objects.

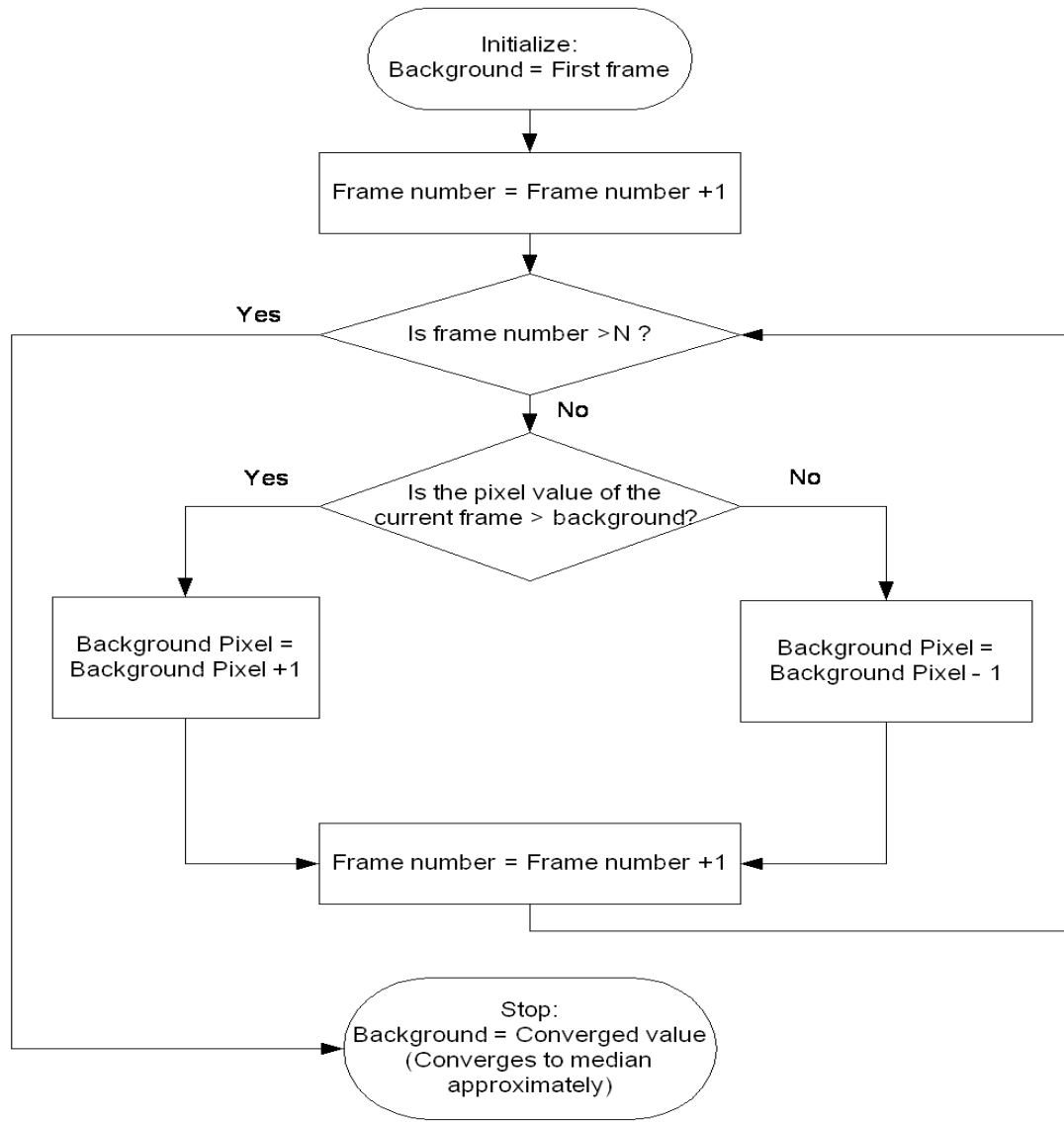


Figure 31: Flowchart summarizing the background determination process

5.3. Classification into pedestrians and non-pedestrians

Classification is carried out if the background subtraction process identifies a new object in the current frame. The most critical aspect of classifying objects into pedestrians and non-pedestrians is the selection and representation of a feature that is unique to pedestrians. In our adopted methodology, the “Ω” shape [31] formed by the head and

shoulders of a human being is used as the feature that distinguishes pedestrians from non-pedestrians. The “ Ω ” shape approach has been implemented for the following reasons

- The “ Ω ” shape remains more or less the same regardless of clothing styles.
- Robustness against shape variation as a person walks versus a full-body based approach.

The frames containing the new object forms the input to an omega shape detector. The detection process is quite different from face detection algorithms since a face need not be clearly visible to detect the “ Ω ” shape.

Shape detectors rely on a numerical shape representation known as a descriptor vector. The number of elements in the descriptor vector is referred to as the dimension of the descriptor. In the adopted methodology, shapes are represented by HoG [38] descriptors. The detector calculates descriptors in a given frame and identifies whether they belong to an omega shape. The location of a window containing the “ Ω ” shape forms the detector output.

5.3.1. Representation of shapes using HoGs

HoGs provide an excellent description for discriminating objects in the presence of cluttered backgrounds under different illuminations [39]. The shape of an object can be characterized using a histogram of edge directions without the knowledge of their precise locations. Edges are pixel locations which have sharp, abrupt changes in brightness values which indicate the presence of a shape. Since a histogram represents the shape and not the precise edge locations and magnitude, the HoG descriptor is robust to minute shape translation and rotation caused by camera movements.

The calculation of an HoG requires an image to be divided into dense overlapping windows of a pre-determined size, as shown in Figure 32. Each image window is further divided into small regions called cells. The HoG descriptor is calculated for each cell. The HoG of the image is represented as the concatenated combination of HoGs of its windows which is a concatenation of histograms of its constituent cells. The HoG calculation process has three phases – edge gradient calculation, histogramming and normalization.



Figure 32: Dense overlapping windows for HoG calculation

5.3.1.1. Edge gradient calculation

The first step in the calculation of HoG is edge gradient computation. Edge gradients identify shape contours by virtue of the differences between neighboring pixels in the east-west and north-south directions. A high difference value indicates the boundary of a shape. If $p(x, y)$ is the pixel value at the location (x, y) , edge values $e(x, y)$ in the x and y directions are represented by

$$e_x(x, y) = p(x + 1, y) - p(x - 1, y)$$

$$e_y(x, y) = p(x, y + 1) - p(x, y - 1)$$

Equation 3: Calculation of edge values

Edges calculated for the x direction include the left and the right neighbors whereas those calculated for the y direction include the neighbors on the top and the bottom. A larger number of surrounding neighbor values may also be considered to detect edges, such as Sobel edge detector [64]. The magnitude and orientation of edges are calculated using Equation 4.

$$Magnitude = \sqrt{(ex^2 + ey^2)}$$

$$Orientation = \tan^{-1}\left(\frac{ey}{ex}\right)$$

Equation 4: Calculation of edge magnitude and orientation

For color images, the edge value is calculated for each of the three channels and the largest value is fixed to be the edge value at that pixel. The overall detector performance is sensitive to the calculation of edges. The original images along with the calculated edges are illustrated in Figure 33.



Figure 33: Original image along with the detected edges

5.3.1.2. Histogramming

The edge orientations of all pixels of a cell are allocated to bins where each bin represents a range of orientation values. The bins are weighted. The weight of a bin is the sum of weights contributed by each orientation it includes. The weight may be the edge

magnitude, the square root of the edge magnitude, a 1/0 value indicating the presence or absence of an edge or any other user-defined measure. In the adopted methodology, the weights are the corresponding edge magnitudes. A collection of weighted bins forms a histogram. The histogram of cells belonging to a window lays the foundation for an HoG descriptor. The HoG descriptor of a window consists of cell histograms arranged according to their physical location in the window. The direction of cell traversal may be horizontal or vertical.

5.3.1.3. Histogram Normalization

Edge strength depends on illumination conditions and the contrast of an object against its background. The same shape applied to different backgrounds and illumination conditions may result to completely different HoGs. Hence, contrast normalization is essential for robustness against lighting and background. Local variations are compensated by grouping cells into larger spatial regions called blocks followed by the normalization of the histogram for a block. The block histogram is the concatenation of histogram of orientations calculated for its constituent cells. Square blocks are used for the sake of simplicity.

$$\#CellsInBlock = \frac{BlockWidth}{CellWidth} * \frac{BlockHeight}{CellHeight}$$

$$BlockHistogramSize = (\#CellsInBlock) * (\#Histogrambins)$$

Equation 5: Definition of block parameters

Each block is normalized individually using $L2$ norm, i.e. the histogram elements are divided by the square root of the sum of squares of all the histogram components in the block. A term ϵ is added to avoid division by zero. For example, if (a, b, c, d) form a descriptor, the normalized descriptor would be

$$\left(\frac{a}{N}, \frac{b}{N}, \frac{c}{N}, \frac{d}{N} \right)$$

$$N = \sqrt{a^2 + b^2 + c^2 + d^2 + \varepsilon^2}$$

Equation 6: Normalized descriptor

A block histogram is shown in Figure 34.

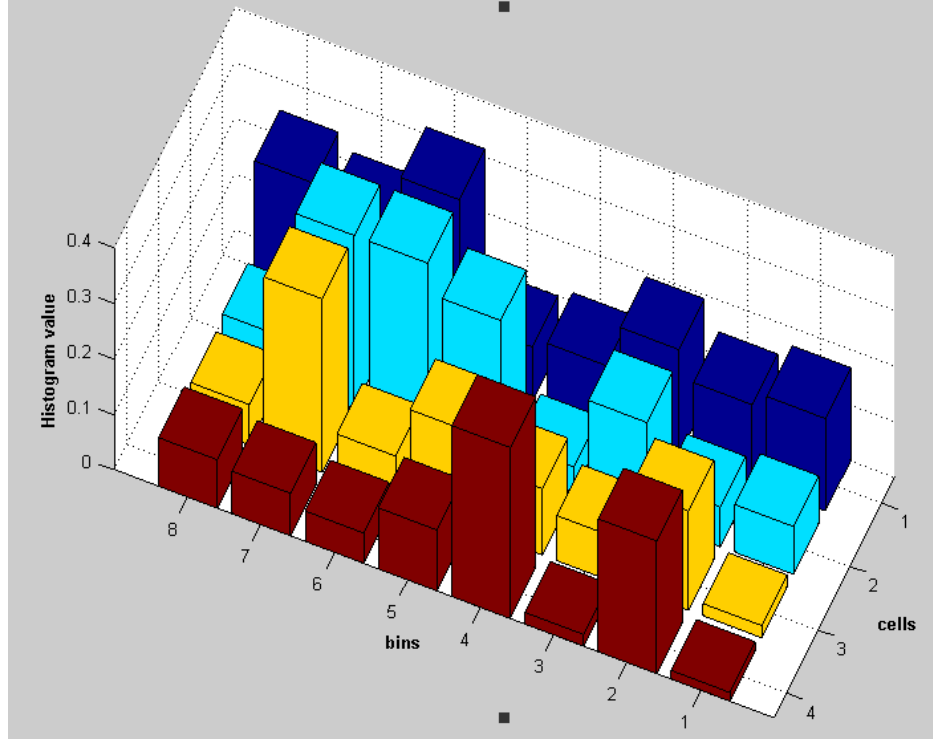


Figure 34: Block histogram with 8 bins and 4 cells

The number of blocks in a window is calculated using the equation

$$\#blocksInWindow = \frac{WindowWidth}{BlockWidth} * \frac{WindowHeight}{BlockHeight}$$

Equation 7: Number of blocks if the blocks do not overlap

Overlapping the blocks improves performance since contrast variation over a larger area is reduced.

$$\#blocksInWindow = \left(\frac{WindowWidth - BlockWidth}{BlockStrideWidth} + 1 \right) * \left(\frac{WindowHeight - BlockHeight}{BlockStrideHeight} + 1 \right)$$

Equation 8: Number of blocks if blocks overlap by an amount called BlockStride

The final descriptor for a window consists of all the normalized descriptors of its constituent blocks.

$$\#DescriptorElements = (BlockHistogramSize) * (\#blocksInWindow)$$

Equation 9: Number of descriptor elements for a window

5.3.1.4. Methodologies to improve HoG

Procedures such as Gaussian weighting and trilinear interpolation were carried out while histogramming [38] for a more accurate representation of shapes using HoGs. The effect of gamma normalization [65] before processing an image has also been evaluated.

Gamma normalization alleviates the effect of non-linear data compression in cameras and tries to reconstruct an image to its original RGB values. The effect of various gamma values on an image is demonstrated in Figure 35.

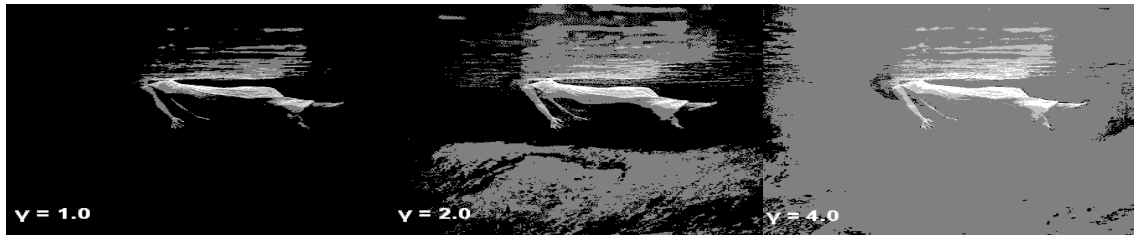


Figure 35: Images with different values of gamma (Source:wikipedia)

Gaussian weighting [38] in histogramming adds additional weight to bins according to the physical location of edge pixels to down-weight pixels near the block borders. This additional weighting process has been found to increase accuracy. The weights are calculated according to the equation

$$p_{new}(x, y) = p_{old}(x, y) * \exp\left(-\left[\frac{(x - \mu_x)^2 + (y - \mu_y)^2}{2\sigma^2}\right]\right)$$

$$\mu_x = BlockWidth * 0.5$$

$$\mu_y = BlockHeight * 0.5$$

$$\sigma = BlockWidth * 0.5$$

Equation 10: Gaussian spatial window applied to each pixel

Trilinear interpolation [38] refers to the distribution of weights assigned to a bin and to neighboring bins. The assignment of weights to bins during histogramming results in low accuracy. An image of the same object using a different camera or a slight change in illumination conditions may change the edge gradient considerably. The orientation bin may itself change in some cases. Bin interpolation mitigates this issue by distributing the weight of an edge orientation between the corresponding bin and the closest neighboring bin. The ratio of distribution depends on the difference between the orientation and bin centers. There may also be situations where minute camera movements may lead to a strong edge gradient being moved to another cell. The two histograms of the same object may look significantly different thereby producing poor classifier performance. Hence, interpolation is required.

Let h be a histogram with inter-bin distance b_x , b_y and b_z in the histogram cube. Values x and y denote spatial position and z denotes the orientation dimension. If a weight w at point (x, y, z) has neighboring points (x_1, y_1) and (x_2, y_2) where $x_1 \leq x < x_2$, trilinear interpolation distributes the weight w into its eight neighboring bin-centers in the space.

$$\begin{aligned}
h(x_1, y_1, z_1) &= w * \left(1 - \frac{x - x_1}{b_x}\right) * \left(1 - \frac{y - y_1}{b_y}\right) * \left(1 - \frac{z - z_1}{b_z}\right) \\
h(x_1, y_1, z_2) &= w * \left(1 - \frac{x - x_1}{b_x}\right) * \left(1 - \frac{y - y_1}{b_y}\right) * \left(\frac{z - z_1}{b_z}\right) \\
h(x_1, y_2, z_1) &= w * \left(1 - \frac{x - x_1}{b_x}\right) * \left(\frac{y - y_1}{b_y}\right) * \left(1 - \frac{z - z_1}{b_z}\right) \\
h(x_1, y_2, z_2) &= w * \left(1 - \frac{x - x_1}{b_x}\right) * \left(\frac{y - y_1}{b_y}\right) * \left(\frac{z - z_1}{b_z}\right) \\
h(x_2, y_1, z_1) &= w * \left(\frac{x - x_1}{b_x}\right) * \left(1 - \frac{y - y_1}{b_y}\right) * \left(1 - \frac{z - z_1}{b_z}\right) \\
h(x_2, y_1, z_2) &= w * \left(\frac{x - x_1}{b_x}\right) * \left(1 - \frac{y - y_1}{b_y}\right) * \left(\frac{z - z_1}{b_z}\right) \\
h(x_2, y_2, z_1) &= w * \left(\frac{x - x_1}{b_x}\right) * \left(\frac{y - y_1}{b_y}\right) * \left(1 - \frac{z - z_1}{b_z}\right) \\
h(x_2, y_2, z_2) &= w * \left(\frac{x - x_1}{b_x}\right) * \left(\frac{y - y_1}{b_y}\right) * \left(\frac{z - z_1}{b_z}\right)
\end{aligned}$$

Equation 11: Trilinear interpolation for pixels

The HoG calculation procedure is summarized in the flowchart shown in Figure 36.

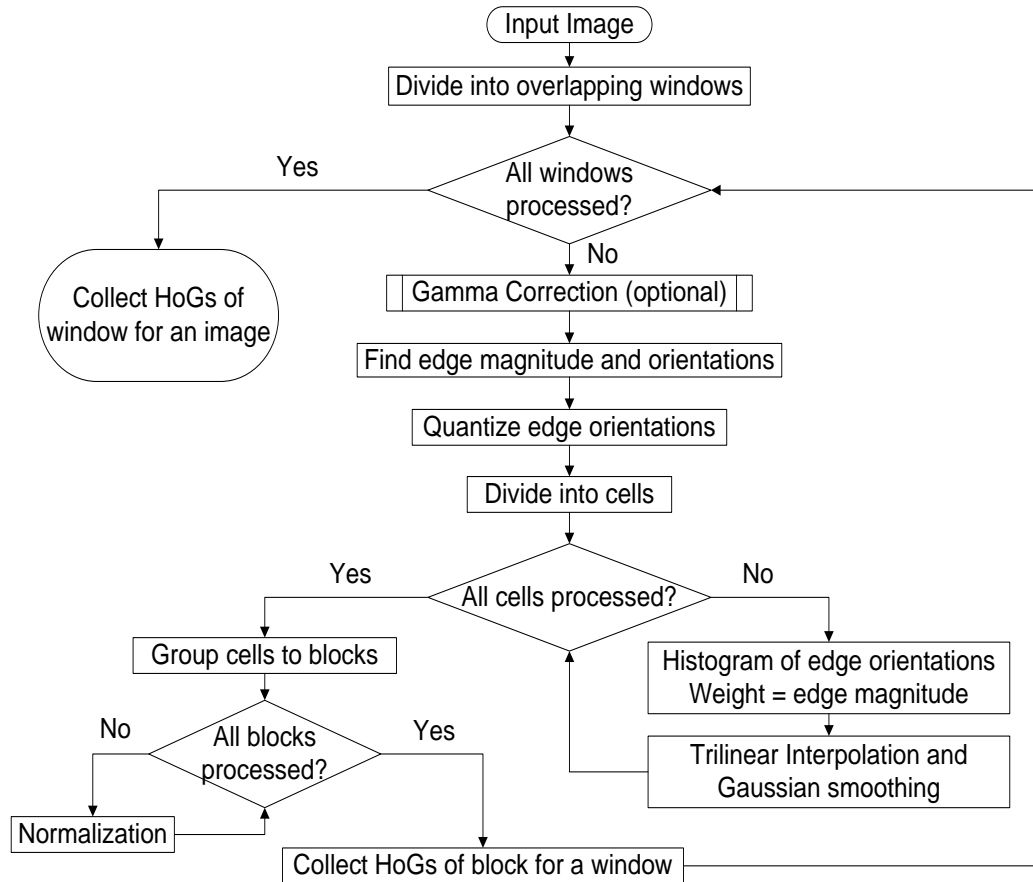


Figure 36: Flowchart representing the process of HoG calculation

The HoG calculated for each window in the image is input to the classifier to determine whether a “ Ω ” shape resides in the window. The HoGs can be calculated at various scales to allow for the accurate detection of pedestrians. The detection and classification algorithm is run on the resized image for each scale. However, multi-scale detection has not been implemented in this thesis.

5.3.2. Classification of shapes

Classification is a decision making process made by the software program to determine whether the input object belongs to a certain category. As discussed in Section 2.2.2.3, the capability to categorize objects is developed through the process of training with examples. A challenge in the training process is to strike the right balance between accuracy and generalization. Accuracy indicates the percentage of samples that a classifier can correctly categorize. Generalization implies that the classifier should be able to correctly categorize an object that is dissimilar to training images. For instance, if the categories are “pedestrians” and “non-pedestrians”, a good classifier should identify most of the pedestrians which are similar to the training images as well as identify a pedestrian who has worn a different style of clothing from the training images. Hence, it is desirable that positive samples (samples containing the desired object) as well as negative samples (samples which do not contain the desired object) be used for classifier training.

5.3.2.1. Support Vector Machines

The support vector machine (SVM) is the classifier used in the adopted methodology. The SVM maps all training samples to a feature space [62], where the two categories are divided by a clear gap that is as wide as possible. In mathematical terms,

given a dataset D consisting of points of the form (\mathbf{x}_i, y_i) where $\mathbf{x}_i \in \mathbf{R}^n$ is the HoG descriptor and y_i is the label, e.g. +1 for a pedestrian and -1 for a non-pedestrian, the training process formulates a function $f(\mathbf{x}_i, \alpha)$ which maps every \mathbf{x}_i to y_i in D . During the classification process, the test descriptor is mapped to the same feature space and a prediction is made regarding the appropriate category based on the location of the mapping. In mathematical terms, for a sample \mathbf{v} , the value of the label y is calculated using $f(\mathbf{v}, \alpha)$.

For descriptors of high dimensions, such as HoG, a linear classifier is preferred. A linear SVM separates p -dimensional vectors using a $p-1$ dimensional hyperplane, which represents the largest separation or margin between the classes. The functionality of SVM is represented in Figure 37. The black points and the white points represent p -dimensional vectors belonging to two separate categories in a feature space, which can be separated by three hyperplanes, H_1 , H_2 and H_3 . H_2 is the best classifier since it separates the two categories with a large separation between the classes.

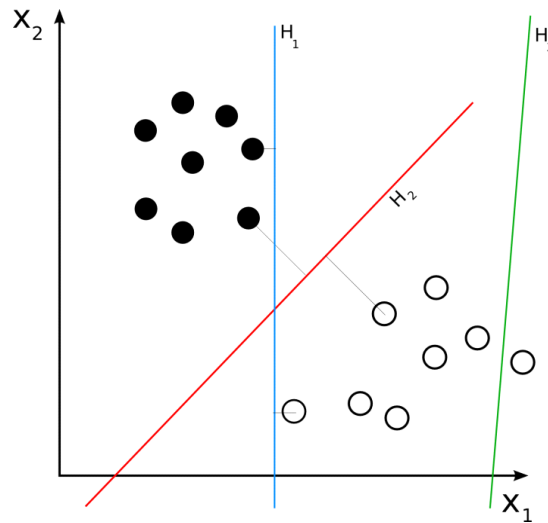


Figure 37: Hyperplanes separating the two categories

The training and classification problem can be formalized as follows [62]. If $\mathbf{w} \cdot \mathbf{x} + b = 0$ defines the hyperplane separating the two categories, where \mathbf{w} represents the slope or direction of inclination of the hyperplane and b is its offset from origin, every training HoG data \mathbf{x}_i shall satisfy the following inequalities

$$\begin{aligned}\mathbf{x}_i \cdot \mathbf{w} + b &\geq 1 + \xi_i; y_i = +1 \\ \mathbf{x}_i \cdot \mathbf{w} + b &\leq -1 + \xi_i; y_i = -1 \\ \xi_i &\geq 0 \forall i\end{aligned}$$

Equation 12: Constraint equation for the SVM hyperplane

The error introduced in training due to the lack of an optimal hyperplane has an upper bound dictated by $\sum \xi_i$. Consider the points where the equality in Equation 12 holds. The equality implies that there are two hyperplanes which are parallel to the optimal hyperplane each defining a boundary for the two categories considered for classification. Hence, the objective of maximizing separation between the two classes reduces to maximizing the distance between the two hyperplanes, as shown in Equation 13.

$$\min \frac{\|\mathbf{w}\|^2}{2} + C \left(\sum_i \xi_i \right)^k$$

Equation 13: Cost function to be minimized

The value of C , the cost factor, is a user-defined parameter. Introducing the Lagrangian factor α_i , Equation 13 is reformulated as

$$\begin{aligned}\max \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ 0 \leq \alpha_i \leq C; \\ \sum_i \alpha_i y_i = 0\end{aligned}$$

Equation 14: Lagrangian of the cost function

The training process involves solving Equation 14 to find $f(\mathbf{x}_i, \alpha)$. Every training HoG point \mathbf{x}_i is associated with α_i . In the solution, the points corresponding to $\alpha_i > 0$ are support vectors, those training HoGs which lie on or between the boundary hyperplanes mentioned above. The concept is illustrated in Figure 38. The value of C indicates the positioning of the bounding hyperplanes. Larger values of C lead to more false detections.

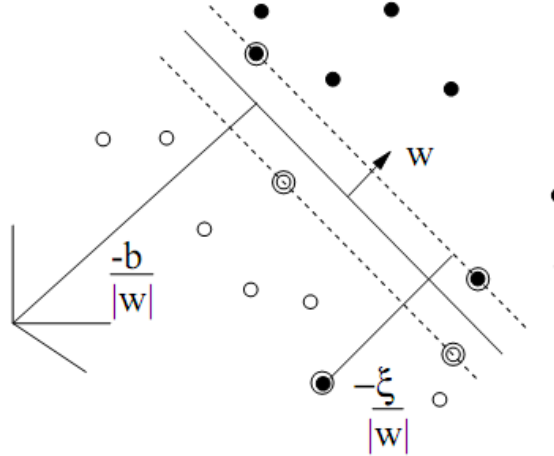


Figure 38: Representation of SVM [65]

White dots and black dots represent two categories of data. The bold line represents the dividing hyperplanes whereas the dotted lines represent the boundary planes.

The points represented by two circles are support vectors.

The classification problem identifies the category of an unknown HoG vector \mathbf{x} to be the sign of $(\mathbf{x} \cdot \mathbf{w} + b)$ where \mathbf{w} and \mathbf{b} are the slope vectors and offset of the hyperplane, respectively.

The data fed to the SVM classifier must be scaled linearly to fall in the range $[-1, 1]$ or $[0, 1]$ using the same scaling factor during training and classification to avoid greater numeric ranges dominating the smaller numeric ranges. The scaling is not carried out in this scenario since the HoGs are already normalized.

5.3.2.2. Data set used for training

One of the major factors affecting the performance of a classifier is the set of images used in training. A variety of training images are available such as the MIT pedestrian dataset [66], the Caltech pedestrian dataset [67] and the INRIA person dataset [68]. The MIT pedestrian dataset consists of upright pedestrians centered in images. The lack of negative samples and a sufficient range of object characteristics render the MIT dataset highly unsuitable for learning purposes. The recently-released Caltech pedestrian dataset consists of pedestrians in a wide variety of poses, some under occlusion. The INRIA person dataset includes numerous pedestrian images for accurate classification, making it the most desirable choice.

The INRIA dataset consists of 614 annotated positive samples containing pedestrians from various locations and 1218 negative samples consisting of roads, landscapes and buildings. The pedestrians are mostly standing, but some images appear in other orientations portrayed against a wide variety of background images including crowds. The pedestrian photographs have been annotated using the PASCAL (Pattern Analysis, Statistical modeling and Computational Analysis Learning organization) annotation format [69]. The PASCAL annotation file consists of relevant information such as the position of objects of interest and the center of the head of each pedestrian in an image.

The 64 x 128 sized images of individual pedestrians from the dataset were generated by a MATLAB script using the information in annotation files. From the 64 x 128 images, portions containing the head and shoulders were cropped manually and resized to a 32 x 32 image. The 32 x 32 window includes a significant amount of context

that aids detection [38]. The left and right reflections of the 32×32 images constitute the positive samples that are fed into the classifier. Figure 39 shows the results of generating positive samples.

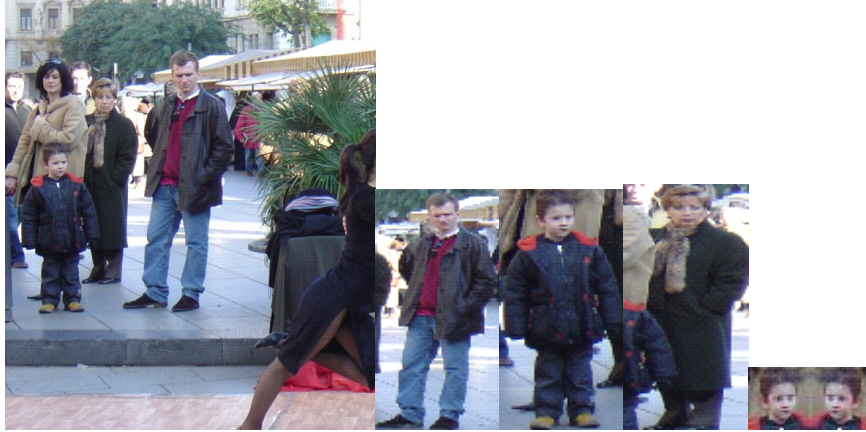


Figure 39: Generation of positive samples for training
Left to right - Original image from the INRIA training dataset [68]. Size=818 x 976;
Extracted objects of interest from the image. Size = 64 x 128; Left right reflection of
the cropped omega shape. Size = 32 x 32

The negative samples for training were generated by randomly choosing 32×32 windows from the negative images in the dataset [38]. Some negative samples are shown in Figure 40.

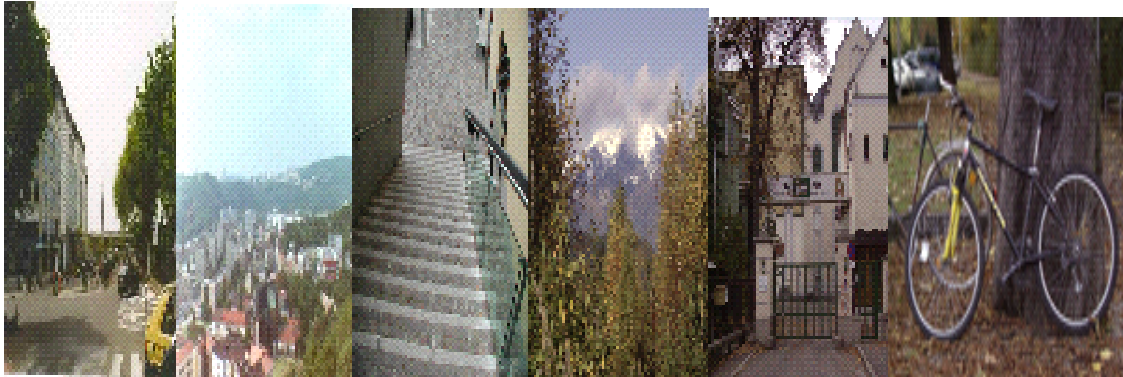


Figure 40: Negative samples in INRIA dataset [68]

After the initial training, all the negative sample images are provided as inputs to the classifier for detecting pedestrians. If the classifier reports the presence of a

pedestrian in a negative sample, the location of the window and the image is noted. Such windows, termed hard examples, along with the initial positive and negative samples are used to retrain the classifier after all negative samples have been processed. The set of hard examples may be uniformly sub-sampled to reduce storage requirements.

5.4. Counting algorithm

One way of counting pedestrians is to carry out detection in every frame and track the pedestrians in subsequent video frames [70]. The task of detecting an object and establishing correspondence between object instances across frames is a complicated process under dense traffic conditions. Preliminary experiments with detector configurations in the Solo Terra have indicated that pedestrians move over a region of 10 pixels wide in approximately half a second as indicated in Figure 41.

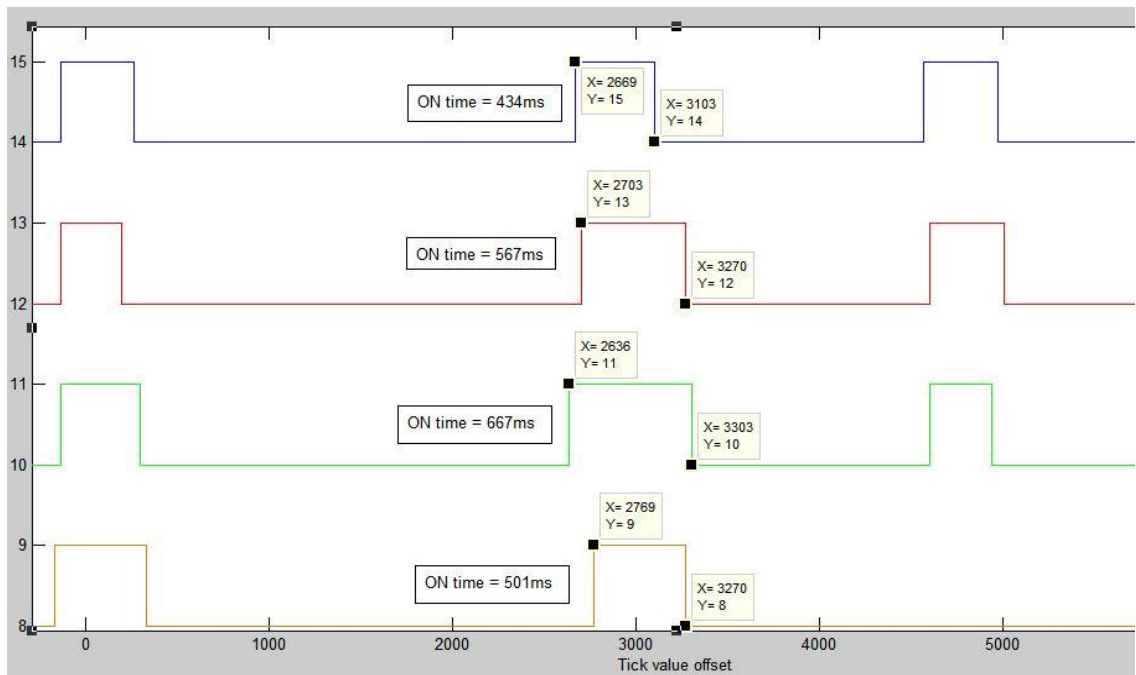


Figure 41: Detector transition times for the Solo Terra when a pedestrian walks across the detector region

Counting can be carried out by detecting pedestrians in periodic snapshots of the sidewalk and processing only those regions where pedestrians are likely to appear. The count is incremented based on the number of detected pedestrians. The user has the option of selecting regions which should be processed at the beginning of the counting process.

5.5. Software libraries

The detection algorithms were implemented in C++ using OpenCV [71] as a main framework component. The OpenCV library is a computer vision library that includes basic computer vision algorithms and machine learning functions. The library, which was originally developed by Intel, is freely available to the public under a BSD license. OpenCV provides APIs to process images and videos. The backbone functions are highly optimized for real-time image processing. The library can be compiled for various operating system platforms using CMake, a cross-compiler. Wrappers for languages such as C#, Python, Ruby and Java have been developed to encourage the adoption of the library by a wide audience.

This thesis implements the described HoG-based algorithm in C++ using OpenCV. The functions for reading and writing video files were developed using the APIs and codec interface functions supported by OpenCV.

CHAPTER 6

INTEGRATED SYSTEM FOR ACCURATE COUNTING

This chapter discusses the integration of the two counting methodologies described in Chapters 4 and 5 to build a robust pedestrian counting system. Several proof-of-concept experiments have been carried out to verify the functionality of the integrated system.

6.1 The need for an integrated approach

The two counting approaches described in Chapters 4 and 5 have advantages as well as disadvantages. The omega detection algorithm performs poorly under occlusion since the head and shoulders of pedestrians must be clearly visible. The Solo Terra approach is robust against occlusion to an extent and is feasible for wide-scale deployment. But there are situations where approximating m detectors to the presence of one pedestrian may not work well. Pedestrians farther away from the camera tend to be missed when the sidewalk is fully occupied along its width. None of the Solo Terra counting algorithms effectively address issues such as overcounts and missed counts. Missed counts are the predominant source of error in the state averaging approach.

The Solo Terra increments the pedestrian count based on OFF-ON-OFF transitions of its detectors. The detectors in the Solo Terra turn ON whenever pixel values in detector regions differ from background pixels. At times, shadows of trees and overhead wires may turn the detectors ON resulting in false counts. Overcounts may also occur when certain pedestrians cover more detector regions than an average-sized pedestrian. Such overcounts are generally minimal in the omega detection approach.

In addition to algorithmic disadvantages, there may be situations where data collected by the Solo Terra is not accurate. Detectors may fail to change state during the autofocus of the camera lens. The count in these situations is not incremented, thus degrading accuracy. These anomalies include blooming and streaking, which are illustrated in Figure 42. Blooming and streaking are caused by extreme exposures, i.e. very bright edges against a relatively dark background [72].

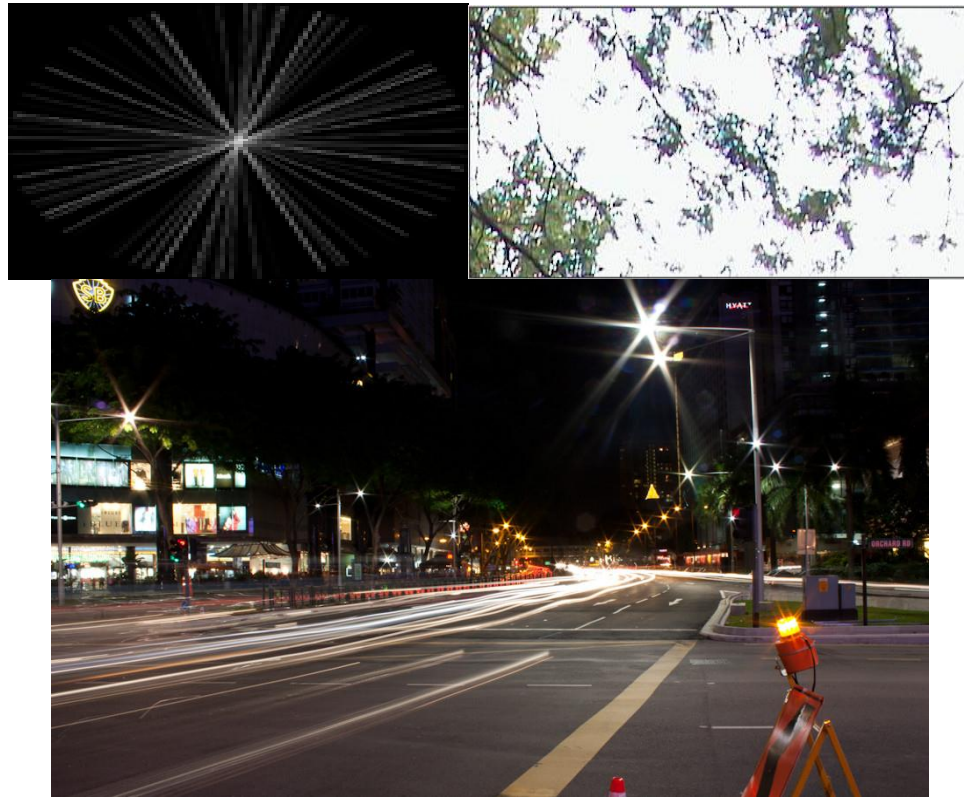


Figure 42: (a) Streaking (b) Blooming (c) Simultaneous blooming and streaking

Streaking refers to light rays appearing in an image due to the diffraction of light when it passes through a narrow aperture. Normally, the effects of diffraction are averaged out, but in dimly lit scenarios vehicle headlights may cause streaking.

Blooming refers to bright spots in an image caused by the limitations of image sensors in a digital camera. The Solo Terra uses Charge Coupled Device or CCD sensors

as image sensors. A CCD is an array of photovoltaic cells that converts light into electrical charge. Each cell is associated with a pixel value which indicates the amount of charge induced by light on a cell. The pixel values are stored as an image. Blooming occurs when a large amount of charge, which exceeds the storage capacity, is induced on a photovoltaic cell. The excess charge leaks to its neighbor pixel or other surrounding pixels brightening or overexposing them in the process. Such pixels reach their maximum value which “brightens” parts of the image. As a result, detectors do not change their states despite the presence of pedestrians, since cells need to discharge before capturing the current image. The discharging process of photovoltaic cells takes a least several seconds which results in undercounts.

A blooming phenomenon was observed during experimentation on a cloudy winter day (Figure 43). The entire image turned bright when a person wearing bright white clothes entered the field of view. Even though this scenario occurs rarely, the count accuracy is highly degraded when it occurs. During experimentation, it was found that the lens settings of the Solo Terra did not mitigate the effects of blooming.



Figure 43: Detectors remaining ON due to blooming resulting in under-counts

Since the Solo Terra increments the pedestrian count based on changes in pixel values, we need an approach that validates the count as it is determined. The omega detection approach processes pixels searching for the shape formed by the head and shoulders of a pedestrian. In general, it cannot be a stand-alone solution for transportation planning since the mounting angle of the camera is not always able to clearly detect the head and shoulders of every pedestrian.

6.2 Experimental framework

The integrated system consists of a Solo Terra camera and a low-cost camera which communicate with a single PC. The PC collects state transition data from the Solo Terra and images from the low cost camera. The two cameras use separate mounting structures to capture videos without disrupting the pedestrian traffic. A low-cost camera is used in conjunction with the Solo Terra to mitigate the effects of continuous refocusing and to overcome the lack of open source codecs which can process the proprietary video recording format of images retrieved from the Solo Terra in detection mode. The Autoscope Software Suite is used on the PC to collect detector information. Microsoft Visual Studio along with OpenCV libraries performs HoG analysis.

A single piece of software written in C++ implements the state averaging algorithm described in Chapter 4 and the omega detection algorithm described in Chapter 5. The linear SVM available in the OpenCV library is used for classification. A count is calculated using detector information obtained from Solo Terra with occasional corrections using the omega detection algorithm. The software monitors the polled state transition data collected by Solo Terra and carries out omega detection only under the following conditions.

- CASE 1: Whenever more than $n = 1.5 * \text{averaging constant}$ detectors are ON simultaneously - A snapshot of the field of view is obtained from the low-cost camera. Regions farther from Solo Terra are scanned for omega shapes. The states of detector regions closer to Solo Terra at that particular instant are considered for the state averaging approach. The final count at that instant is the sum of counts obtained from both approaches. Missed counts due to occlusion can be addressed.
- CASE 2: Whenever more than $n = 1.5 * \text{averaging constant}$ detectors are ON simultaneously for more than four seconds – This condition indicates that the Solo Terra is refocusing itself. The count is solely incremented based on the omega detection approach until the detectors turn OFF.

In all other cases, the count is incremented based on the state averaging approach. The count at each instant along with a timestamp is dumped into a text file which can be processed at a later point of time for statistics collection. In the longer term, the processing will be performed in real time. Overcounts due to shadows of moving branches may be avoided if sufficient detectors are triggered to start the omega detection process.

The two cameras need to be synchronized in time so that the correct frame is captured by the low-cost camera to modify counts. The synchronization can be carried out at the start of experiment as follows. The system is deployed at the test location in a manner similar to the one shown in Figure 44. The Solo Terra is loaded with a detector configuration and is subsequently polled for detector states. At the same time, a small rectangular region close to the detector regions is scanned for omega shapes. The region is located in images in a video stream captured using the low-cost camera. The initial tick

field value in Solo Terra is noted. The tick value is incremented every millisecond. When the Solo Terra detectors are triggered by a pedestrian crossing the region, the frame number of the video stream captured by the low-cost camera is observed. The frame number of the first detection by omega detection software in the small window is also noted. The difference between the two frame numbers creates an offset used for Case 1. Depending on the offset difference, the integrated software must wait before processing the frame. The frame processing rate, scanning window size and HoG parameters for omega detection in Case 2 are fixed based on Section 5.4.

Detailed field experiments with the integrated approach have not yet been performed. The deployed low-cost camera streams images with a frame size of 320 x 240, hence a mounting height of 20 feet will likely be sufficient in future experimentation to prevent occlusion. The envisioned framework is shown in Figure 44.

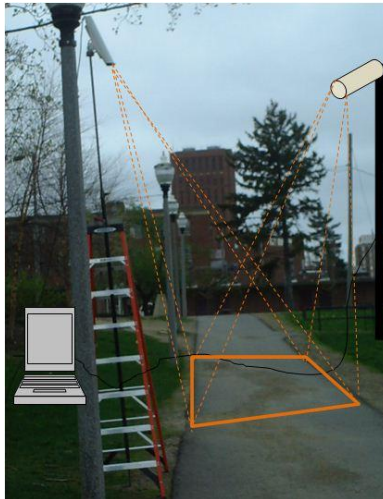


Figure 44: Experimental framework for the integrated approach

CHAPTER 7

EXPERIMENTS AND RESULTS

This chapter details experiments that were conducted for evaluating the two pedestrian counting approaches. The vision-based shape detection approach was determined to be beneficial in detecting pedestrians located in various parts of an image.

7.1. Experiments using the Autoscope Solo Terra

Selecting a suitable location for experiments was an important aspect of the project. The suitability was determined by the presence of a power outlet near the location, reasonable pedestrian traffic and walkway widths similar to a sidewalk. The majority of experiments were conducted under good weather conditions to avoid shadows and variations in lighting. Experiments were conducted for 35 days at two locations in University of Massachusetts, Amherst - the sidewalk near Engineering Lab II and the Marcus Hall ramp (Figure 45). Both locations are straight walkways of approximately 2.5 meters in width, the typical width of a sidewalk. Pedestrian flow rates usually fall between 5 and 15 persons/min, and can be as high as 100 persons/min during peak periods. The dense pedestrian traffic in opposite directions emulated a crowded pedestrian sidewalk in urban areas. The Solo Terra camera was placed near the sidewalk on the mounting structure described in Chapter 4. The height of the camera was fixed at 15 feet.



Figure 45: Test locations for counting pedestrians using Solo Terra

Preliminary experiments used two adjacent detection zones located on a walkway. Later experiments used two non-overlapping count detectors. The background refresh rate was fixed at 90 seconds. The following observations were made from the experiments.

- Detection zones supported by the Configuration Wizard counted pedestrians moving only in one direction.
- Detectors counted pedestrians moving in both directions with 75% accuracy. The system performance degraded under heavy traffic.
- Count accuracy was highly sensitive to shadows irrespective of pedestrian density.
- Approximately 15% of the pedestrians were counted twice.
- Detectors partially covered by building shadows were ON despite the constant state of the background during the refresh interval.

- Objects must cover at least one-fourth of a detector region to cause an OFF-ON transition.



Figure 46: Detectors counting pedestrians

This list of issues prompted the use of multiple detectors as described in Sections 4.3 and 4.4. Some of the coding for these experiments was performed by Chaoqun (Enzo) Chia in MATLAB. Accuracy results for different pedestrian flow rates i.e. the number of pedestrians walking per minute was calculated on different days. Pedestrian counting accuracy is defined as:

$$\text{Accuracy} = \left(1 - \frac{\#mistakes}{Groundtruth} \right)$$

Equation 15: Definition of accuracy

Counting mistakes fall into three categories – overcounts, undercounts, and missed counts. An overcount is a situation where a single pedestrian increments the count more than once. In most cases, the count is incremented twice. An undercount is the scenario where the count is incremented only once when two or more pedestrians walk across the detector region. A missed count is a special case of undercount where the

count is not incremented, even for a single pedestrian. The term *#mistakes* is the sum of extra counts due to overcounts and the missed counts falling under the other two categories. The ground truth is the number of manually-counted pedestrians. This count can be verified in some cases using recorded videos of the experiment. For example, if one pedestrian increments the count by three, another pedestrian is missed and two others increment the count only once, then

$$Groundtruth = 1 + 1 + 2 = 4$$

$$\#mistakes = 2(\text{extra count due to overcounts}) + 1(\text{undercount error}) + 1(\text{missed pedestrian}) = 4$$

7.1.1. Results from the state averaging approach

In an initial experiment at UMass, pedestrian counts were evaluated for a range of averaging constants, m , for pedestrians walking along a walkway. If m is defined as the averaging constant and N is the total number of detectors turning ON at a particular instant, the number of pedestrians at that instant is given by

$$\#Pedestrians = \frac{N}{m}$$

The corresponding accuracy values for different values of m are tabulated in Table 6 for seven video segments. Videos 1, 2, 3 and 4 use one camera mounting angle and videos 5, 6 and 7 use a second mounting angle.

Table 6: Accuracy for each of the averaging parameter m

#Video	$m = 10$	$m = 11$	$m = 12$	$m = 13$
1	66	83	91	100
2	68	81	90	100
3	70	81	92	100
4	82	93	95	89
5	91	100	91	84
6	94	96	88	81
7	93	96	88	81

It can be concluded that $m=11$ and $m=13$ work best for the detector configurations. For a fixed mounting height of 15 feet, the best and worst case accuracies over six trials of dense pedestrian traffic, each of duration six minutes, are given in Table 7. The large under-counts in the worst case scenario are attributed to refocusing when detector status was not updated. Hence, the averaging algorithm which is based on counting detector transitions did not increment the count.

Table 7: Best and worst case results for 2 x 15 configuration with uniform spacing

Ground truth	Count from algo.	Avg. pedestrian flow rate	Minimum #pedestrians in a minute	Maximum #pedestrians in a minute	Extra count	Mis ses	Accuracy
146	147	29.2 ped/min	6	73	4	3	95.2 %
164	139	27.8 ped/min	14	51	2	26	82.9%

The accuracy for different pedestrian flow rates with an averaging constant $m=11$ is summarized in Table 8. The accuracy is independent of pedestrian flow rate since the count depends on the position of pedestrians rather than pedestrian density.

Table 8: Accuracy for different pedestrian densities

#Video	Ped. density	Ground truth	Under-counts	Over-counts	Accuracy
1	18.5	9	1	0	89
2	55	35	6	2	77
3	66.7	20	4	0	80
4	90	30	5	1	80
5	81.1	21	4	0	81
6	87.3	16	2	0	87

A detector configuration of five columns of 15 detectors each, as shown in Figure 47, yielded an average accuracy around 81% over four trials.

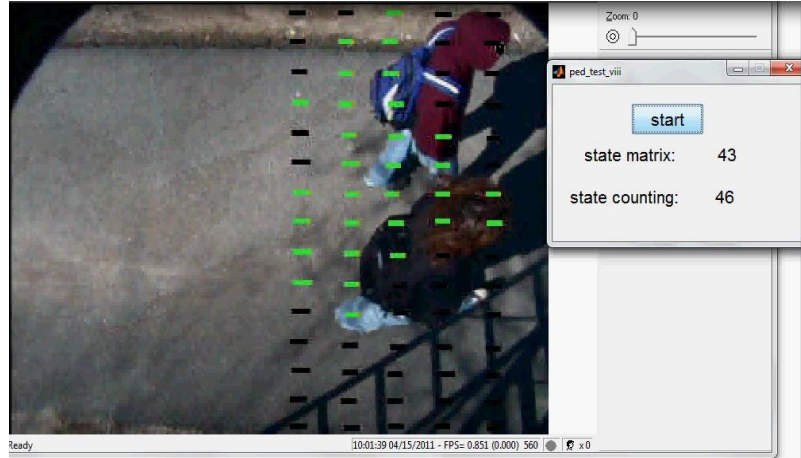


Figure 47: A five column detector configuration for averaging approach

During experimentation it was found that pedestrians that are closer to the camera occupy less image area than pedestrians who are farther from the camera. Hence, detectors were configured in two columns with non-uniform spacing (Figure 48).



Figure 48: Detector configuration with non-uniform spacing

The best and worst accuracies achieved for this configuration during 6 six-minute trials of dense pedestrian traffic are tabulated in Table 9.

Table 9: Best and worst case results for a 2 x 15 configuration with non-uniform spacing

Ground truth	Count from algo.	Avg. pedestrian flow rate	Minimum #pedestrians in a minute	Maximum #pedestrians in a minute	Extra count	Mis ses	Accuracy
160	148	26.67 ped/min	0	89	4	11	90.62 %
161	145	32.2 ped/min	0	78	4	20	85.10%

The two-column configuration of 15 detectors with uniform spacing gave the best accuracy values for the averaging approach. Undercount was the major source of error. Shadows did not degrade accuracy under dense pedestrian traffic because of occlusion. However, under low traffic conditions, overcounts occurred for the shadow of every pedestrian crossing the detector region.

7.1.2. Results from the state matrix approach

This approach was formulated to address scenarios where sufficient detectors do not turn ON for averaging. In all conducted real-time trials, the average accuracy for this approach was found to be 40%, primarily due to under-counting. Hence, detailed analyses of the results were not performed and the approach was abandoned.

7.1.3. Results from the M of N approach

Accuracy was measured for different amount of overlaps and different values of M and N. It was found that a significant number of undercounts occurred when $M \geq 0.75N$ and overlap ≤ 1 . Overcounts contributed to the error when $M \leq 0.75N$ or overlap = 0.5N. Configurations with varying numbers of detectors were evaluated. It was found that a configuration of 15 detectors, each of size 9 x 2 pixels in a column as illustrated in Figure 28, gave the best performance with overlap = 0.25N when M= 0.75N. The approach is highly sensitive to mounting height and angle. Hence, measured accuracies

ranged from as low as 45% to as high as 99%. The lack of consistency and robustness for irregular pedestrian traffic movements discourage the M of N approach's use in wide scale deployment. It can also be concluded from experiments that uneven group sizes, for instance three groups with values of N=6, 5, 6, gave better consistency than evenly divided groups, although the average accuracy is higher for evenly-sized groups. Pedestrians farther away from the camera occupy more space in the image when compared to pedestrians closer to the camera. The resultant accuracies from various detector configurations are tabulated below. The detector configuration of 11 and 13 detectors in a column was tested for two videos. The results from experiments with a single column of 15 detectors were averaged over six videos, each of duration 6 minutes.

Table 10: Results of M of N approach

#detectors	N	M	overlap	Average flow rate	Average Accuracy
11	5	4	2	5.667 ped/min	73.52%
13	5	4	1	25.2 ped/min	80.39%
15	5	3	0	17.89 ped/min	79.67%
15	5	4	0	17.89 ped/min	80.24%
15	6,6,5	4,4,3	1	17.89 ped/min	74.45%
15	6,5,6	4,3,4	1	17.89 ped/min	70.13%
15	6	4	2	17.89 ped/min	82.67%
15	6	4	3	17.89 ped/min	55.33%
15	4	3	0	17.89 ped/min	77.00%
15	4	3	1	17.89 ped/min	61.67%

To gauge the effect of detector sizes on count accuracy, different columns of detectors were formulated as a single configuration, as shown in Figure 49. Detectors of dimensions ranging from 2 to 10 pixels were used to provide redundancy for the multiple detector M of N approach. Larger detectors gave rise to over-counts. Highly dense

detector configurations were very sensitive to the values of M and N and the amount of region overlap.

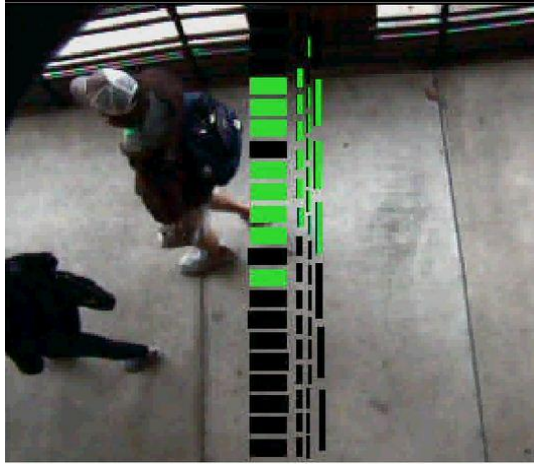


Figure 49: A detector configuration to find the effect of sizes on pedestrian count accuracy

7.1.4. Effect of background refresh rate

During counting, background refresh rates for the count detectors were fixed to 20, 60, 90 and 600 seconds. Refresh rates of 60 seconds and higher did not affect count accuracy. The low refresh rate of 20 seconds led to random detector state transitions although accuracy was not degraded since the count was only updated only when a certain minimum number of detectors were turned ON.

7.2. Experiments using the low-cost camera

A Sony NSC-GC1 camera was used for vision-based shape detection. The camera records video in MPEG4 format with a frame size of 640 x 480 and a frame rate of 30 frames per second. The camera supports the streaming of 320 x 240 video frames using a 32-bit Windows operating system.

To reduce false detections and ensure real-time operation, only a section of each frame containing the sidewalk was scanned for pedestrian heads and shoulders. Overlapping or non-overlapping rectangular scanning regions were specified prior to the experiment. The first frame of streaming video was used to select the processing area in subsequent frames. The selected region coordinates were rounded off to nearest multiples of the window size and subsequently processed sequentially in software. Figure 50 illustrates detections regions (blue rectangles) and identified pedestrians (green rectangles).



Figure 50: Scanning for omega shapes in user-selected frame regions

During experimentation, it was found that pedestrians cross a 10 pixel wide region in under a second (Figure 41). Hence, two seconds of activity signified by sixty frames were buffered to determine the background, as described in Section 5.2. The threshold which determines foreground from background was varied depending on lighting conditions. Under perfect lighting conditions, a threshold of five gave accurate results. Currently, the background is not updated. The results of background subtraction are shown in Figure 51.



Figure 51: Original frame; Results of background subtraction - Identified foreground is white in color, the black pixels indicate the background

It was determined through experiments that color and context information aid detection. A camera mounting height of 20 feet ensures that a pedestrian's head and shoulders fit into a 32 x 32 pixel sized window. The entry of a pedestrian into a window is signified by change in more than 250 of the 1024 pixels in the window (32 x 32). Hence, the HoG of a window in the frame is calculated if more than 250 pixels in the scanned window change values. This approach ensures the real-time operation of the omega detection algorithm.

7.2.1. Evaluation of HoG parameters

An exhaustive sweep over HoG parameters was carried out on the MIT Pedestrian Dataset [66] and the optimum values of all parameters were determined. Pre-processing has shown a negative impact on the classification process during experiments. Accuracy degraded by 8% when gamma processing was carried out on the data set. The edge gradients were calculated as described in Section 5.3.1.1. The maximum orientation of gradients was fixed at 180 and 360 degrees. A 360 degree range with eight histogramming bins gave minimal false detections. Varying weight types such as magnitude, square magnitude and square root of edge gradient magnitude were evaluated

in the sweep. The magnitude and square magnitude votes gave minimal false detections. The effects of cell traversal direction on histogramming were also evaluated. A horizontal traversing direction resulted in a longer training duration. However, accuracy remained the same for both directions. Block normalization using L1 norm, L2 norm, L2-Hys norm and L1-root norm [38] were carried out. L2 norm and L2-Hys norm gave the best trade-off between misses and false detections. L2-Hys norm is the L2-norm described in Section 5.3.1 clipped to the range 0.2 to 1. L2-Hys norm gave a marginal improvement of 3% over L2-norm in terms of false detections. High block-overlap (2 pixels) increased the execution time with negligible improvements in accuracy. The block and cell size were fixed to 8 x 8 and 4 x 4 pixels respectively. The optimum HoG parameters [38] for detecting the “Ω” shape are listed in Table 11.

Table 11: Optimum values of parameters for "Ω" detection

Parameter	Value
Window Size	32 x 32 pixels
Sliding distance for windows	8 x 8 pixels
Block Size	8 x 8 pixels
Cell Size	4 x 4 pixels
Number of bins in the histogram	8
Range of orientations	0-360 degrees
Width of the Gaussian Spatial Window	4 pixels
Gamma Correction	No
Trilinear Interpolation	Yes
Type of norm	L2_Hys
Type of vote	Magnitude

7.2.2. Experiments with SVM classifier

The classification software was trained using a total of 15234 images consisting of 2054 positive samples, 12180 negative samples and 1000 hard examples. Each sample was a 32 x 32 image cropped from the INRIA Person Dataset [68]. To generate negative

samples, ten 32 x 32 windows were randomly selected from each negative image in the dataset. The classification software was initially tested on the MIT pedestrian dataset [66]. A result is shown in Figure 52.



Figure 52: Omega detection results on the MIT Pedestrian Dataset [66]

The accuracy of classifying a shape using a linear SVM as a pedestrian or non-pedestrian depends on the hyper-plane that separates HoG points of the two categories. A suitable cost factor aids in finding a hyperplane, as mentioned in Section 5.3.2.1. The cost factor tries to strike a balance between generalizing and over-fitting training data. Increasing the cost factor of SVM tends to increase generalization performance. This leads to a higher number of positive detections and an increase in the number of false detections. A decrease in cost factor implies over-fitting. This issue translates to misses when an omega shape which is dissimilar to training samples in shape and color is presented to the classifier. The cost factor for training the SVM was fixed at $C=0.01$. The values of the categories are fixed at +1 which implies the presence of a pedestrian and -1 which implies the absence of a pedestrian, resulting in a b value which is nearly zero. The weight vector \mathbf{w} has the same size as the HoG vector.

A trade-off exists between false detections and misses. Reducing false detections increases misses and vice versa. Hence, during training, weights were given to each category to reduce false detections and misses. The hyper-plane formulated during

training is shifted towards the category with larger weight. The category with lower weight can then be favored during decision-making. Equal weights were given to both categories initially. After the evaluation of negative training samples, hard training samples were generated. Training was carried out using the same weights over the positive, negative and hard samples.

To increase accuracy, a Gaussian SVM available in the OpenCV library was used for experimentation. A Gaussian SVM separates the HoGs using curves rather than a hyper-plane, as shown in Figure 53. The red and blue circles represent the training points belonging to two separate categories. After training, classification was carried out on the MIT Pedestrian Dataset [66]. Overall, the approach provided negligible improvement at the cost of extensive processing time, discouraging exploration in this direction.

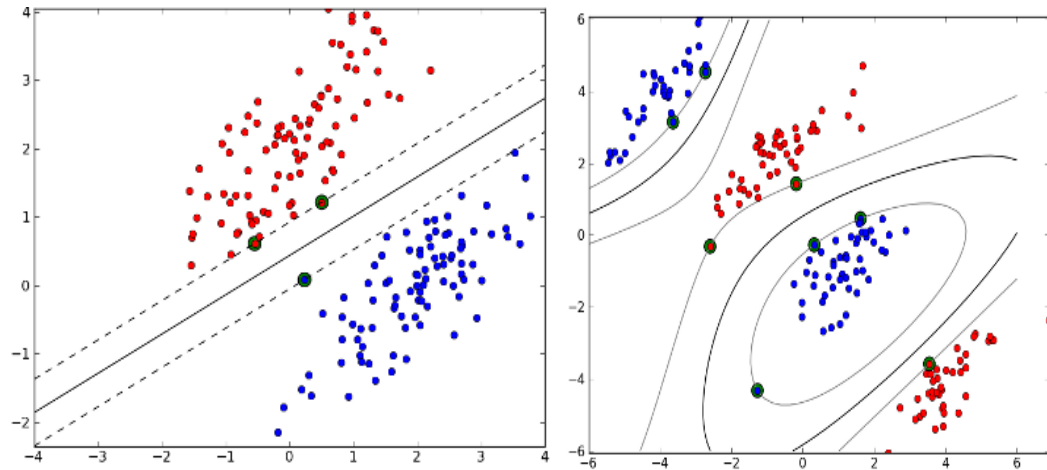


Figure 53: (a) Linear SVM where a hyperplane divides the two categories (b) Gaussian SVM where curves separate the two categories

Image: <http://www.mblondel.org/journal/2010/09/19/support-vector-machines-in-python/>

A simple classifier based on a naïve comparison of two histograms cannot be considered for real-time shape detection since the histograms of omega shapes and non-omega shapes have high dimensions and similar profiles, as shown in Figure 54.

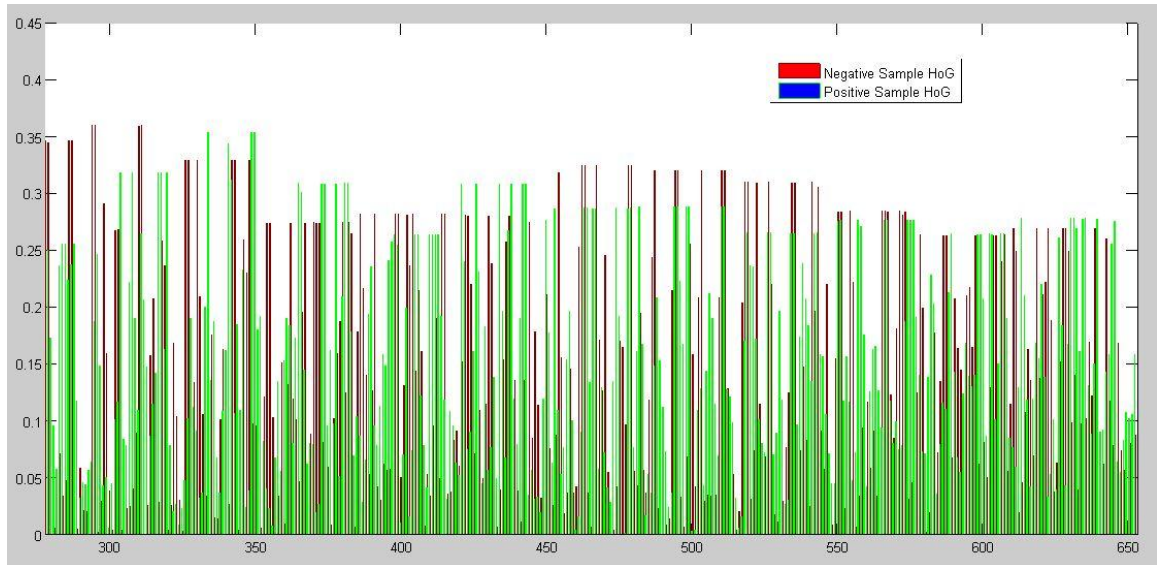


Figure 54: Histogram of a positive sample and a non-omega shape
Maroon represents negative HoG whereas green represents positive HoG

Classifier software verification was followed by experiments using videos. The test videos contained people walking in arbitrary directions. It was observed that the head and shoulders of a pedestrian must be clearly visible and must fit into a window of a size between 28 x 28 to 34 x 34 pixels to enable detection. Rectangles appear around the shape to indicate that the object has been classified as a pedestrian by the software. A snapshot of such a video is shown in Figure 55.



Figure 55: Pedestrian detection using the low-cost camera.
The detected pedestrians are marked by a rectangle around their head

7.2.3. Software performance issues

This section describes the various issues encountered while executing the omega detection software. Occlusion remains an issue to be addressed.

7.2.3.1. Multiple rectangles around detected shapes

Rectangles around a shape indicate a positive detection. The number of rectangles around the detected object depends on the similarity of the object to the training images, the amount of window overlap and the distance of the object from the camera. Initial experiments resulted in multiple rectangles around a shape as demonstrated in Figure 56. Since the count is incremented based on the number of rectangles, it was necessary to ensure the identification of a single rectangle for every detected pedestrian. Rectangle grouping algorithms available in OpenCV [74] and the mean shift algorithm used by Dalal et al. [38] were considered to mitigate the issue of multiple rectangles. The OpenCV [74] algorithm merges groups of rectangles that lie within a user-defined distance. The position of the final rectangle is the average of all rectangles lying within that range. The mean shift algorithm [38] merges rectangles that lie within a user-defined distance by considering the SVM scores for each rectangular region.



Figure 56: Results of rectangle grouping
Left to right - Multiple rectangles around an object when it is close to the camera,
Multiple rectangles around an object when it is close to the camera , Results of
OpenCV algorithm [74], Results of algorithm proposed by Dalal [38]

Rectangle grouping algorithms require a minimum number of rectangles around the detected object, necessitating multi-scale detection. Detection at various image scales requires significant data storage which may hinder real-time processing. For pedestrian counting a camera mounting height of 20 feet may be necessary. The minimum number of rectangles required for grouping may not be available since pedestrians may not be sufficiently close to the camera. A mounting height which ensures that the head and shoulders lie in a 32 x 32 window helps to mitigate the issue. Simple window overlap reduction approaches applied during scanning and weight addition in the SVM training process were implemented to avoid multiple rectangles.

7.2.3.2. Execution time

The initial execution time for the HoG algorithm was found to be as high as twenty minutes for a one-minute video. The software was executed on a 2.66GHz Intel Core2Duo CPU using an Ubuntu operating system. The execution time was measured using “gprof”, a performance profiler obtained from GNU. Execution time measurements from gprof are tabulated in Table 13.

Table 12: Execution time reported by gprof for initial software

Function	% of time spent for execution
Classification	97.76
Edge detection	1.51
HoG calculation	0.90
Others	0.05

The code was restructured and the frame processing rate was set to one frame per second to reduce execution time. The classification process was found to be the performance bottleneck. Subsequently, the initial implementation was modified to carry out classification only when a sufficient number of pixels differed from the background

in the user selected regions of the field of view. The scanning window distance was fixed at 8. The OpenCV library [71] calculated the inner product of support vectors and alpha values for each window as described in Section 5.3.2.1. The result was multiplied with the test HoG vector. A typical linear SVM would consider at least 1000 support vectors, each of the same length as a HoG descriptor. The linear SVM [71] carried out computations on raw HoG test data and look-up tables of weight vectors were generated at the beginning of the detection process to enhance processing [73]. Tables were also formulated for storing certain HoG parameters. The total processing time for a one-minute video with a processing rate of one frame per second is one minute for the modified software. A pedestrian detection accuracy of 80% was achieved over a set of six benchmark videos, each of duration two minutes. The benchmark videos were collected from walkways at the University of Massachusetts, Amherst. The percentage of time spent in each function as reported by gprof is reported in Table 13.

Table 13: Execution time reported by gprof

Function	% of time spent for execution
Gradient Calculation	51.26
Histogramming	45.13
Classification	3.56
Others	0.05

7.3. Results of integrating the two approaches

Preliminary experiments were carried out to evaluate the functionality of the integrated approach. Two image conditions were identified. One condition scans for omega shapes when more than certain number of Solo Terra detectors is ON during a one second period. The second condition scans for omega shapes when more than certain number of detectors remains ON for a pre-determined period. The Solo Terra

was configured with two columns of 15 count detectors each as described in Section 7.1.1. The first demarcated region encompassed regions farther away from the camera as indicated by the blue rectangle in Figure 57, which is covered by half of the detectors. The second demarcated region covered the entire area around the detectors covered by the red rectangle in Figure 57.



Figure 57: Regions demarcated for searching for omega shapes.

Five scenarios were considered while verifying the software.

- A group of five people walking along a walkway
- A group of four people walking along a walkway
- A group of three people walking along a walkway
- Multiple groups of two people walking along a walkway
- A single person carrying a box.

The detector activation counts determined by the Solo Terra were dumped into a text file. It was found that the HoG approach can correct the Solo Terra count determined using state averaging when 4 or more people walk across the detector region in a single line as shown in Figure 58. In all other cases, count was only determined by the state

averaging approach. Accuracy results for the conducted experiment are tabulated in Table 14.

Table 14: Preliminary results for the integrated approach

Ground truth	Count from Solo Terra approach	Count from HoG approach	Count from integrated approach
34	29	27	32

Scenarios where the omega detection algorithm is invoked but does not increment the count are illustrated in Figure 59. Figure 60 shows when the algorithm is not invoked in the integrated approach.



Figure 58: Scenario where count gets corrected by the HoG approach in the integrated system



Figure 59: Scenarios which invoked omega detection algorithm but did not increment count



Figure 60: Scenario where omega detection does not get invoked

The accuracy for the integrated approach is highly sensitive to the region demarcated for the omega shape search and the amount of window overlap. A slight variation of 6 pixels may affect the accuracy in the presence of dense pedestrian traffic. The window overlap parameter allows for a tradeoff between accuracy and execution time. A sliding distance of 4 pixels, leading to an overlap of 28 pixels, increases accuracy, but also increases processing and execution time to as high as 10 seconds for a single frame. A sliding distance of 8 pixels over a small-sized region reduces accuracy at the cost of real-time execution. Hence, the sliding distance can be fixed to be 6-8 pixels depending on the size of the processed region.



Figure 61: Effect of window sliding distance on accuracy. (a) Sliding distance = 8 pixels, (b) Sliding distance = 6 pixels

During experiments, the processed area was set to approximately 100 x 150 pixels and the sliding distance was fixed at 8 pixels. The processed area is divided into 32 x 32 windows and scanned for HoGs.

CHAPTER 8

CONCLUSION AND FUTURE WORK

The thesis has implemented an automated system to count pedestrians on a sidewalk. The main objective of the work is to aid transportation planning. Two approaches were considered for the detection and counting of pedestrians. The first approach uses a Autoscope Solo Terra device, a widely deployed traffic camera. This approach uses detection zones to identify pedestrians passing through specific regions of an image. The second approach employs a low-cost digital camera to acquire videos for vision-based shape detection. This approach detects pedestrians based on images of their head and shoulders. The Autoscope Solo Terra approach was found to provide over 85% pedestrian counting accuracy in many experiments. The vision-based shape detection approach provided 80% accuracy, although stringent camera mounting requirements may limit its wide-scale deployment. In addition, the approach performs poorly when pedestrians are occluded. In a final experiment, the two approaches were integrated together to form a single system that can effectively count pedestrians. A preliminary prototype of the integration software has been developed and evaluated.

The methodology adopted for counting pedestrians may be extended to count cyclists in roadside bike lanes. In the future, a complete system including both pedestrians and cyclists can be developed and refined.

Disclaimer

This document was prepared in cooperation with the Massachusetts Department of Transportation and Public Works, Office of Transportation Planning, and the United States Department of Transportation, Federal Highway Administration. The contents of

this thesis reflect the views of the author who is responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official view or policies of the Massachusetts Department of Transportation and Public Works of the Federal Highway Administration.

BIBLIOGRAPHY

- [1] (2010, Nov.) [Online]. <http://www.walkinginfo.org/pedsafe/crashstats.cfm>
- [2] (2010, Nov.) [Online]. http://www.econolite.com/docs/vehicle-detection/autoscope_solo_terra_data_sheet.pdf
- [3] T. Gandhi and M. M. Trivedi, "Computer Vision and Machine Learning for Enhancing Pedestrian Safety," in *Computational Intelligence in Automotive Applications*, Springer Berlin, 2008, pp. 79-102.
- [4] D. Noyce and R. Dharmaraju, "An Evaluation of Technologies for Automated Detection and Classification of Pedestrians and Bicyclists," Massachusetts Highway Department, 2002.
- [5] (2010, Feb.) [Online]. <http://www.mssedco.com/>
- [6] F. Bu, R. Ragland, M. Diogenes, and R. Roesel, "Estimating pedestrian accident exposure," CalTrans, Automated Pedestrian Counting Devices Report Task Order 6211.
- [7] F. Bu and C. Chan, "Literature review of pedestrian detection technologies and sensor survey," Institute of Transportation Studies, University of California at Berkeley., Mid-term Report for Experimental Transit Vehicle Platform for Pedestrian Detection, California PATH Task Order 5200.
- [8] D. Noyce, A. Gajendranand, and R. Dharmaraju. (2006) Development of a Bicycle and Pedestrian Detection and Classification Algorithm for Active-Infrared Overhead Vehicle Imaging Sensors. In Transportation Research Board 85th Annual Meeting, Transportation Research Board, Washington , D.C. CD-ROM.
- [9] T. B. Moeslund and E. Granum, "Multiple cues used in model-based human motion capture," in *IEEE International Conference on Automatic Face and Gesture Recognition*, 2000, p. 362–367.
- [10] G. Sexton, X. Zhang, G. Redpath, and D. Greaves, "Advances in Automated Pedestrian Counting," in *European Convention on Security and Detection*, 1995.
- [11] I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: Who? When? Where? What? A real time system for detecting and tracking people," in *International Conference on Face and Gesture Recognition*, 1998, p. 222–227.
- [12] T. Zhao and R. Nevatia, "Bayesian human segmentation in crowded situations," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, p. 459–466, 2003.

- [13] M. Harville, G. Gordon, and J. Woodfill, "Adaptive background subtraction using colour and depth," *IEEE International Conference on Image Processing*, vol. 3, p. 90–93, 2001.
- [14] H. Nanda, C. BenAbdelkader, and L. S. Davis, "Modeling pedestrian shapes for outlier detection: A neural net based approach," in *IEEE Intelligent Vehicle Symposium*, 2003, p. 428–433.
- [15] K. Kim, D. Harwood, and L. S. Davis, "Background updating for visual surveillance," in *International Symposium on Visual Computing*, 2005, p. 337–346.
- [16] T. Kohonen, "Learning vector quantization," *Neural Networks*, vol. I, pp. 3-16, 1988.
- [17] L. Zhao and C. Thorpe, "Stereo and Neural Network-Based Pedestrian Detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. I, pp. 148-154, Sep. 2000.
- [18] A. Howard, L. H. Matthies, A. Huertas, M. Bajracharya, and A. Rankin, "'Detecting Pedestrians with Stereo Vision: Safe Operation of Autonomous Ground Vehicles in Dynamic Environments,'" in *International Symposium of Robotics Research*, November 2007.
- [19] H. Nanda and L. Davis, "Probabilistic Template Based Pedestrian Detection in Infrared Videos," in *Procs. IEEE Intelligent Vehicles. Symposium*, Paris, June 2002.
- [20] Y. Fang, K. Yamada, Y. Ninomiya, B. Horn, and I. Masaki, "A Shape-Independent-Method for Pedestrian Detection with Far Infrared-images," *Special Issue on "In-Vehicle Computer Vision Systems" of the IEEE Transactions on Vehicular Technology*, vol. 53, no. 6, pp. 1679-1697, Nov. 2004.
- [21] K. C. Fuerstenberg, K. Dietmayer, and V. Willhoeft, "Pedestrian Recognition in Urban Traffic using a Vehicle based Multilayer Laser Scanner," in *IEEE Intelligent Vehicles Symposium*, pp. 31-35.
- [22] (2010, Feb.) [Online]. http://www.cambridgeconsultants.com/news_pr108.html
- [23] (2010, Feb.) [Online]. http://www.nissan-global.com/EN/NEWS/2007/_STORY/070417-01-e.html
- [24] C. Curio, J. Edelbrunner, T. Kalinke, C. Tzomakas, and W. V. Seelen, "Walking pedestrian recognition," *IEEE Transactions on Intelligent Transport Systems*, vol. I, pp. 155-163, 2000.
- [25] H. Mori and S. Yasutomi, "A method for discriminating pedestrians based on rhythm," in *IEEE/RSG Intl Conf. on Intelligent Robots and Systems*, 1994.

- [26] R. Cutler and L. Davis, "Robust Real-Time Periodic Motion Detection, Analysis and Applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 781-796, 2000.
- [27] T.-H. Hou and M.-D. Pern, "A computer vision-based shape-classification system using image projection and a neural network," *The International Journal of Advanced Manufacturing Technology*, vol. 15, no. 11, pp. 843-850, 1999.
- [28] L. Snidaro, C. Micheloni, and C. Chiavedale, "Video security for ambient intelligence," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 35, pp. 133-144, 2005.
- [29] J. K. Aggarwal and Q. Cai, "Tracking human motion using multiple cameras," *International Conference on Pattern Recognition*, vol. 3, pp. 68-72, 1996.
- [30] J. Park, Y. Luo, H. Wang, and Y. L. Murphey, "Pedestrian Detection by Modeling Local Convex Shape Features," in *International Conference on Pattern Recognition*, 2008.
- [31] A. Broggi, M. Bertozzi, A. Fascioli, and M. Sechi, "Shape-based pedestrian detection," in *IEEE Intelligent Vehicles Symposium*, 2000, pp. 215-220.
- [32] K. Mikolajczyk, C. Schmid, and A. Zisserman, "Human detection based on a probabilistic assembly of robust part detectors," in *Proc. 2004 European Conference on Computer Vision*, 2004, pp. 69-81.
- [33] J. Garcia, N. D. V. Lobo, M. Shah, and J. Feinstein, "Automatic detection of heads in colored images," in *Canadian Conference on Computer and Robot Vision*, 2005, pp. 276-281.
- [34] P. Kelly, "Pedestrian Detection and tracking using stereo vision techniques," PhD thesis, Dublin City University, Ireland, 2000.
- [35] (2010, Feb.) [Online]. <http://www.goldennumber.net/>
- [36] C. Papageorgiou, T. Evgeniou, and T. Poggio, "A trainable pedestrian detection system," in *IEEE Intelligent Vehicles Symposium*, 1998, pp. 241-246.
- [37] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio, "Pedestrian detection using wavelet templates," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 193-199.
- [38] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Computer Society conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886-893, 2005.

- [39] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: A Benchmark," in *IEEE Computer Vision and Pattern Recognition*, Miami, 2009.
- [40] J. Mantas, "Methodologies in pattern recognition and image analysis - a brief survey," *Pattern Recognition*, pp. 1-6, 1987.
- [41] D. M. Gavrila and V. Philomin, "Real-time object detection for smart vehicles," in *IEEE International Conference on Computer Vision*, 1999, pp. 87-93.
- [42] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2005, pp. 878-885.
- [43] D. Beymer and K. Konolige, "Real-time tracking of multiple people using continuous detection," in *International Conference on Computer Vision*, 1999.
- [44] A. Micilotta, E. Ong, and R. Bowden, "Detection and tracking of humans by probabilistic body part assembly," in *British Machine Vision Conference*, 2005, pp. 419-428.
- [45] T. Darrell, G. Gordon, M. Harville, and J. Woodfill, "Integrated person tracking using stereo, color, and pattern detection," in *IEEE Conf. Computer Vision and Pattern Recognition*, 1998, pp. 601-608.
- [46] P. Remagnino, et al., "An integrated traffic and pedestrian model based vision system," in *British Machine Vision Conference*, 1999, pp. 380-389.
- [47] S. S. Intille, J. W. Davis, and A. F. Bobick, "Real-time closed-world tracking," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 697-703.
- [48] J. I. Park and S. Inoue, "Hierarchical depth mapping from multiple cameras," in *International Conference on Image Analysis and Processing*, 1997, pp. 685-692.
- [49] R. Munoz-Salinas, E. Aguirre, M. Garcia-Silvente, and A. Gonzalez. (2005) People detection and tracking through stereo vision for human-robot interaction. Lectures Notes on Artificial Intelligence, pages 337–346.
- [50] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, pp. 153-161, 2005.
- [51] B. D. Lucas, "Generalized Image Matching by the Method of Differences," Robotics Institute, Carnegie Mellon University Doctoral Dissertation, 1984.

- [52] H. Mori, N. M. Charkari, and T. Matsushita, "On-line vehicle and pedestrian detections based on sign pattern," *IEEE Transactions on Industrial Electronics*, vol. 41, pp. 384-391, 1994.
- [53] N. T. Siebel and S. J. Maybank, "Fusion of multiple tracking algorithms for robust people tracking," in *European Conference on Computer Vision*, 2002, pp. 373-387.
- [54] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Hilton Head, SC, June 2002, pp. 142-149.
- [55] M. Isard and A. Blake, "Condensation - conditional density propagation for visual tracking," in *International Journal of Computer Vision*, 1998.
- [56] P. Beardsley and E. Bourrat, "Wheelchair detection using stereo vision," MERL, Technical report, 2002.
- [57] V. Bhuvaneshwar and P. Mirchandani, "Real-Time detection of crossing Pedestrians for Traffic-Adaptive Signal Control," in *IEEE Intelligent Transportation Systems Conference*, Washington D.C, 2004.
- [58] (2010, Nov.) [Online]. <http://www.arm.com/markets/home/autoscope-solo-terra-video-detection-system.php>
- [59] (2010, Nov.) [Online]. http://autoscope.com/products/dl/AS_TerraInterfacePanel.pdf
- [60] (2010, Nov.) [Online]. <http://www.trafficproducts.com/PDF/TIP%20data%20sheet.pdf>
- [61] "Autoscope Online Help Manual".
- [62] J. C. Burghes, "A tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, no. 2, pp. 121-167, 1998.
- [63] C. P. Schofield and N. J. B. McFarlane, "Segmentation and Tracking of Piglets in Images," *Machine Vision and Applications*, vol. 8, pp. 187-193, 1995.
- [64] (2010, Nov.) [Online]. <http://www.pages.drexel.edu/~weg22/edge.html>
- [65] C. A. Poynton, *Digital Video and HDTV: Algorithms and Interfaces*. Morgan Kaufmann, 2003.
- [66] M. P. Dataset. (2010, Nov.) [Online]. <http://cbcl.mit.edu/cbcl/software-datasets/PedestrianData.html>
- [67] (2010, Nov.) CalTech Pedestrian Dataset. [Online]. http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/

- [68] (2010, Nov.) INRIA Person Dataset. [Online]. <http://pascal.inrialpes.fr/data/human/>
- [69] (2010, Nov.) [Online]. <http://pascallin.ecs.soton.ac.uk/challenges/VOC/>
- [70] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," *ACM Journal of Computing Surveys*, Dec. 2000.
- [71] (2010, Nov) [Online]. <http://opencv.willowgarage.com/wiki/>
- [72] (2011, April) [Online]. <http://www.digital-photo-secrets.com/tip/1175/how-to-avoid-blooming/>
- [73] F.M. Khan, M.G Arnold, W.M. Pottenger, "Hardware-based support vector machine classification in logarithmic number systems," *IEEE International Symposium on Circuits and Systems*, May 2005.